STIC Search Report
EIC 2100

STIC Database Tracking Number: 166843

TO: William Wood **Location: RND 5B15** **Art Unit : 2193** **Monday, September 26, 2005** **Case Serial Number: 09/977717**	**From: Geoffrey St. Leger** **Location: EIC 2100** **Randolph-4B31** **Phone: 23450** **geoffrey.stleger@uspto.gov**

Search Notes

Dear Examiner Wood,

Attached please find the results of your search request for application 09/977717. I searched Dialog's patent files and technical databases.

Please let me know if you have any questions.

Regards,

Geoffrey St. Leger
4B31/x23540

166843

STIC EIC 2100
Search Request Form

Today's Date: 9/26/05

What date would you like to use to limit the search?
Priority Date: 10/12/01 Other:

Name W. Wood

AU 2193 Examiner # 79020

Room # 5B15 Phone 2-3736

Serial # ~~09/~~ 09/977,717

Format for Search Results (Circle One):
(PAPER) DISK EMAIL

Where have you searched so far?
(USP) DWPI (EPO) (JPO) ACM IBM TDB
IEEE INSPEC SPI Other _____

Is this a "Fast & Focused" Search Request? (Circle One) (YES) NO
A "Fast & Focused" Search is completed in 2-3 hours (maximum). The search must be on a very specific topic and meet certain criteria. The criteria are posted in EIC2100 and on the EIC2100 NPL Web Page at http://ptoweb/patents/stic/stic-tc2100.htm.

What is the topic, novelty, motivation, utility, or other specific details defining the desired focus of this search? Please include the concepts, synonyms, keywords, acronyms, definitions, strategies, and anything else that helps to describe the topic. Please attach a copy of the abstract, background, brief summary, pertinent claims and any citations of relevant art you have found.

* Needed:

A function/method/procedure which inherits from another function/method/procedure. Thus one function is derived from another.

STIC Searcher Geoffrey St-Leger Phone 23510

Date picked up 9/26/5 Date Completed 9/26/5

sira Search and Information Resources Administration

US005522072A

United States Patent [19]

De Bruler

[11] Patent Number: 5,522,072

[45] Date of Patent: May 28, 1996

[54] **ARRANGEMENT FOR EFFICIENTLY TRANSFERRING PROGRAM EXECUTION BETWEEN SUBPROGRAMS**

[75] Inventor: **Dennis L. De Bruler**, Downers Grove, Ill.

[73] Assignee: **AT&T Corp.**, Murray Hill, N.J.

[21] Appl. No.: **577,427**

[22] Filed: **Sep. 4, 1990**

[51] **Int. Cl.**[6] **G06F 9/46**

[52] **U.S. Cl.** **395/700**; 364/281.7; 364/281.6; 364/247.7; 364/965.4; 364/957.6; 364/DIG. 1; 364/265.6; 364/944.6; 364/967.4; 395/650

[58] **Field of Search** ..... 364/DIG. 1, DIG. 2; 395/650

[56] **References Cited**

U.S. PATENT DOCUMENTS

3,348,211	10/1967	Ghiron	364/200
3,568,158	3/1971	Haller et al.	340/172.5
3,794,980	2/1974	Cogar et al.	395/375
4,152,761	5/1979	Louie	364/200
4,297,743	10/1981	Appell et al.	364/200
4,530,049	7/1985	Zee	364/200
4,755,935	7/1988	Davis et al.	395/250
4,777,588	10/1988	Case et al.	395/800
4,847,755	7/1989	Morrison et al.	364/200
4,887,215	12/1989	Kumagai et al.	364/431.04
4,945,480	7/1990	Clark et al.	364/200
4,951,194	8/1990	Bradley et al.	364/200

FOREIGN PATENT DOCUMENTS

1216429	8/1989	Japan	G06F 9/42

OTHER PUBLICATIONS

C Programming Tools for i960™ Microprocessor Family, Intel brochure, entire brochure.

S. McGeady, "A Programmer's View of the 80960 Architecture", Thirty–Fourth IEEE Computer Society International Conference *Compcon Spring 89* Digest of Papers, (Feb. 27–Mar. 3, 1989), pp. 4–9.

S. McGeady, "Inside Intel's i960CA Superscalar Processor", *Microprocessors and Microsystems*, vol. 14, No. 6 (Jul./Aug. 1990), pp. 385–396.

J. Letz & J. Slingwine, *Living With RISC: Software Issues in the Regulus Architecture*, Proceedings of the 1987 IEEE International Conference on Computer Design: VLSI in Computers & Processors, Oct. 1987, 549–557.

D. P. Ryan, *Intel's 80960: An Architecture Optimized for Embedded Control*, IEEE Micro, No. 3, Jun. 1988, 63–76.

80960KB Programmer's Reference Manual, Intel, Inc. (1989) pp. 4–1 to 4–15.

M. J. Bach, *The Design of the UNIX® Operating System*, Prentice–Hall, Inc. (1986), pp. 159 to 171.

UNIX System User's Manual, Release 5.0, Bell Laboratories, Inc. (Jun. 1982) p. SETJMP (3C).

"Maxicomputing in Microspace", *UNIX™ Microsystem WE® 32100 Microprocessor Information Manual*, AT&T, (Jan. 1985), pp. 3–28 to 3–32, 4–16 to 4–23, and 5–37 to 5–43.

Y. Chu (ed.), *High–Level Language Computer Architecture*, Academic Press, Inc. (1975) pp. 77 to 81.

C. P. Pfleeger, *Machine Organization*, John Wiley & Sons, Inc. (1982) pp. 123 to 139.

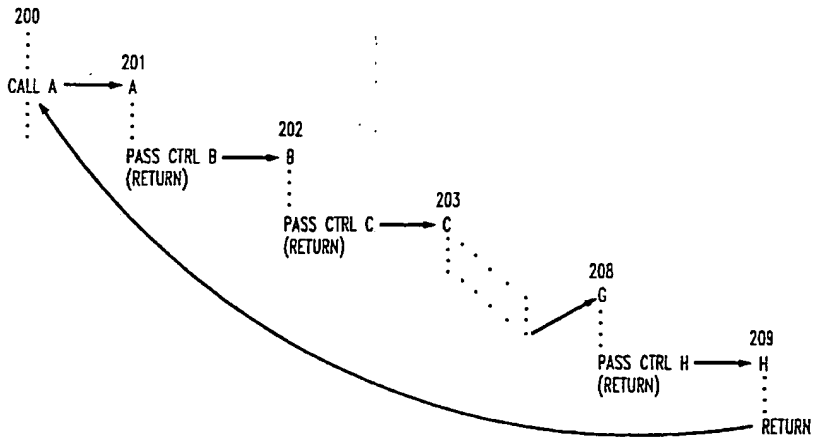
H. Lorin, *Introduction to Computer Architecture and Organization*, John Wiley & Sons, Inc. (1982) pp. 95 to 106.

Primary Examiner—Kevin A. Kriess
Assistant Examiner—Lucien Toplu
Attorney, Agent, or Firm—David Volejnicek

[57] **ABSTRACT**

An arrangement called PASS CONTROL (FIG. 11) is used in combination with a conventional RETURN statement as a substitute for a conventional CALL-and-RETURN subprogram invocation sequence (FIG. 2), and effects a return from a whole series of subprogram invocations directly to the subprogram that initiated the series without intervening returns to the subprograms that made the intermediate invocations in the series. The arrangement uses the conventional execution stack (114) to effect the series of invocations and the return therefrom (FIGS. 12 –14). The subprograms that are invoked by the series of invocations share an execution stack frame (1620). Both a compiler arrangement and an application program execution arrangement for effecting PASS CONTROL functionality are disclosed.

42 Claims, 10 Drawing Sheets



ARRANGEMENT FOR EFFICIENTLY TRANSFERRING PROGRAM EXECUTION BETWEEN SUBPROGRAMS

TECHNICAL FIELD

The invention relates to arrangements for linking the execution of subprograms in computers.

BACKGROUND OF THE INVENTION

A subprogram is a program which invokes the execution of another program and whose execution is eventually returned to from an invoked program, or a program whose execution is invoked by another program and which ends by returning execution to another program, such as the program that invoked it. Of course, a program can be both an invoking and an invoked subprogram. Subprograms are also known by other names, such as routines, subroutines, functions, and procedures. The term "subprogram" as used herein encompasses all such variations.

Subprograms are extensively used in structured, or modular, programming—a commonly-used programming technique that breaks a task up into a sequence of sub-tasks. Structured programming can result in long chains of subprograms, wherein an initial, or main, subprogram invokes subprogram A, which in turn invokes subprogram B, which invokes subprogram C, and so on. Such chains of subprograms are characteristic of certain types of application programs, such as protocol handlers, transaction handlers, database query servers, and call processing systems. These types of application programs are also characterized by throughput constraints. All other things being equal, throughput is directly proportional to the speed of program execution. Consequently, it is important that the invocations of subprograms and the returns from those invocations be executable as quickly as possible.

Of course, one way of speeding up program execution is to use a faster computer. But computer speed is typically directly proportional to computer cost, and hence this approach is costly. Furthermore, technology invariably sets practical limits to the speed of program execution that can be achieved at any time in the continuum of technological development. Consequently, it is important that the invocation of subprograms and the returns from those invocations be implemented as efficiently as possible in order to achieve the fastest possible program execution with a given computer or computer technology.

The traditional and ubiquitous manner of implementing an invocation of a subprogram and a return from that invocation is the CALL and the RETURN statements. These high-level instructions work together with a stack—a last-in, first-out memory structure. Each subprogram that has not completed execution has a frame of information on the stack. Stored in the stack frame is information such as arguments or parameters, local variables, return values, and other information associated with the subprogram. The CALL statement results in the storage of the context of the calling subprogram in a stack frame, and creation of a stack frame for the called subprogram. The context includes information such as general registers contents, instruction pointer contents, and a condition code. The RETURN statement results in deletion of the stack frame of the returning, previously-called, subprogram from the stack, and the restoration of the processor to the stored context of the returned-to, previously-calling, subprogram.

All of this manipulation of memory contents and processor state is time-consuming, and hence detracts from system throughput. Furthermore, in a long chain of subprogram calls, the stack can grow to occupy a significant area of memory, thereby reducing the amount of memory available for other uses. Aside from the obvious memory limitations that this can impose, it can also detract from system throughput by increasing the frequency of occurrence of certain activities, such as swapping of pages to and from main memory.

In view of these disadvantages of the conventional subprogram call-and-return arrangement, attempts have been made to improve upon it, but with limited success.

One known assembler and link editor combination employs a feature known as "leaf proc", which causes the link editor to change a standard call instruction into a branch-and-link (BAL) instruction. The BAL instruction leaves an address of the following (in terms of compilation, as opposed to execution, order) instruction stored in a predetermined off-stack location, such as a general-purpose register, and then performs a conventional branch operation to the target instruction as specified by the operand of the BAL instruction. The state of the stack remains unchanged thereby. A return is then accomplished with a branch instruction, whose operand is the general-purpose register into which the BAL instruction had stored the return address. However, the BAL instruction is limited in use for calls to subprograms that are written in assembly language, that do not call other subprograms, including themselves, and that do not require more than a few general-purpose registers.

The UNIX® operating system employs two pairs of features known as "setjmp" and "longjmp". One pair is implemented as algorithms of the operating system itself, while the other pair is implemented as library functions in the user interface to the operating system.

At the operating system kernel level, the "setjmp" algorithm performs a context switch, but instead of storing the saved context on the stack, saves it in the new process memory area which contains process control information that need be accessed only in the context of that process (the u area). Execution then continues in the context of the old process. When the kernel wishes to resume the context it had saved, it uses the "longjmp" algorithm, which restores the saved context from the u area. This technique is confined to implementations of operating system kernels that include the notion of processes. It is a system-level function available only to the operating system kernel, and is not accessible for use by application programmers.

At the user interface level, the "setjmp" function performs a context switch and saves the old context on the stack, but stores pointers to that context in a predetermined off-stack location. Thereafter, the "longjmp" function, when given the address of the predetermined off-stack location as a parameter, restores and returns to the stored context. While the "longjmp" function can save much of the overhead of the conventional RETURN statement, the "longjmp" function does nothing to reduce the overhead of the conventional CALL statement, such as CPU instruction cycles and stack memory consumption. Also, any subprogram that contains a longjmp instruction must have an ancestor subprogram (a preceding subprogram in the chain of subprogram invocations) that executed a setjmp instruction. Consequently, a subprogram that uses the "longjmp" function cannot be called "normally", as opposed to a subprogram that can be called without restriction from any context. Furthermore, a setjmp cannot be used within a recursive invocation chain (a

sequence of subprogram invocations wherein the last invocation of the sequence invokes the subprogram that made the first invocation in the sequence, thereby forming a loop), though the setjmp can be used by an ancestor subprogram of the recursive chain.

Finally, certain known interpreters make use of a feature known as "tail recursion". Usable only in a self-referential recursive function, and usable only when the last action that the recursive function performs before returning is to call itself, this feature suppresses generation of a stack frame for the called iteration of the function and instead, reuses the frame of the calling iteration of the function. While effective in eliminating much of the overhead of a conventional CALL-and-RETURN statement sequence, the "tail recursion" feature is limited only to the self-referential recursive function calls, and therefore has limited applicability and usefulness.

SUMMARY OF THE INVENTION

This invention is directed to solving these and other disadvantages of the prior art. According to the invention, an arrangement, illustratively referred to herein as PASS CONTROL, effects a return from a whole series of invocations of different subprograms directly to the subprogram that initiated the series, without intervening returns to the subprograms that made the intermediate invocations in the series. According to a first aspect of the invention, presented is a method and an arrangement for executing a series of subprogram invocations of a plurality of different subprograms. The series commences with a first invocation, followed by at least one intermediate invocation, and thereafter ends with a last invocation. During execution of the subprograms, in response to each invocation in the series of a subprogram by an invoking subprogram, execution of the invoking subprogram ceases and execution of the invoked subprogram commences. Further in response to the first invocation in the series, a pointer to an instruction at which the execution of the invoking subprogram that made the first invocation is stored on the execution stack. Illustratively, in contrast, an instruction pointer is not stored in response to any other invocation in the series. Then, in response to a return from a subprogram that was invoked by the last invocation in the series, execution of the returned subprogram ceases, the stored instruction pointer is retrieved from the execution stack, and execution of the subprogram that made the first invocation in the series is resumed at the instruction pointed to by the retrieved instruction pointer. This branch of execution from the last-invoked subprogram to the subprogram that initiated the series of invocations is direct or immediate, without a meantime resumption of execution of any subprogram that made an intermediate invocation in the series.

Because a return from the last-invoked subprogram is effected directly to the subprogram that initiated the series of invocations, the time that is conventionally consumed in both performing the individual returns between each calling and called subprogram, and in manipulating the execution stack in order to effect the individual returns, is saved. System throughput and performance are markedly improved in modular implementations as a consequence. Yet the method and arrangement use the conventional execution stack in order to effect the series of invocations and the return therefrom, as opposed to using an exceptional, special, off-stack mechanism, as is done in some prior art arrangements. Consequently, unlike in the prior art, a subprogram can be invoked normally in all circumstances. That is, in a single program, the same one subprogram can be invoked by either or both of the standard CALL arrangement and the PASS CONTROL arrangement, and without the execution system having to keep track of, or adjust its operation in consequence of, the type of invocation used.

Preferably, all of the subprograms that are invoked by the invocations in the series share an execution context, thereby further reducing the time spent on effecting invocations and returns therefrom. In response to the first invocation in the series, the execution context of the invoking subprogram that made the first invocation is stored on the execution stack. In contrast, execution context is not saved in response to any other invocation in the series. Rather, the existing context is retained and shared. Then, in response to the return from the subprogram that was invoked by the last invocation in the series, execution context is restored to the one context that is stored on the stack. Of course, this restoration is done directly, without the conventional intervening restoration of the contexts of subprograms that made intermediate invocations in the series.

Furthermore, preferably the amount of memory consumed by the execution stack during the series of invocations is significantly reduced and limited, by causing all of the subprograms that are invoked by the series of invocations to share a single stack frame, as opposed to creating a new stack frame for each invoked subprogram. In response to the first invocation in the series, a stack frame for the invoked subprogram is created on the execution stack, and the instruction pointer and execution context of the invoking subprogram are stored therein. In contrast, a stack frame is not created in response to any other invocation in the series, nor are an instruction pointer and an execution context stored on the stack frame in response thereto. Information, e.g., local and temporary variables, for each invoked subprogram may be stored in the shared stack frame in one of two ways: either in an area of the stack frame that is shared by all of the invoked subprograms, or in the one of a plurality of such areas that each invoked subprogram has allocated for its own use in an expandable shared frame. Parameters are passed by each invoking subprogram to each invoked subprogram either through general registers, or through a shared parameters area of a stack frame of the subprogram that initiated the series of invocations and that precedes the shared stack frame on the execution stack. Then, in response to the return from the last-invoked subprogram, the execution context and instruction pointer are restored from the current stack frame (the shared stack frame, in this case), and the entire current stack frame is deleted from the execution stack, as is done by any return invocation.

According to a second aspect of the invention, the functionality characterized above is brought about by a novel compiler arrangement for compiling a source program that comprises a plurality of different source subprograms that are made up of source code statements including subprogram invocation statements and subprogram return statements. Specifically, the source program includes a series of statements of invocation of a plurality of different source subprograms. Except for the first invocation statement in the series, the statements in the series are each included in a source subprogram that is invoked by a preceding invocation statement in the series. The series commences with the first invocation statement followed by at least one intermediate invocation statement and thereafter ends with a last invocation statement. The compiler arrangement compiles the source program into an object program that comprises a plurality of object subprograms made up of object instructions. In response to encountering in a source subprogram a

statement invoking a source subprogram, the compiler arrangement generates an instruction to branch from execution of an object subprogram compiled from the source subprogram that includes the invoking statement, to execution of an object subprogram compiled from the invoked source subprogram. In response to encountering in a source subprogram a CALL statement which is the first invocation in the series, the compiler arrangement generates instructions to store on the execution stack a pointer to an instruction at which is to resume execution of an object subprogram compiled from the source subprogram that includes the first invocation statement. Illustratively, in contrast, the compiler arrangement does not generate instructions to store an instruction pointer in response to encountering any of the PASS CONTROL statements which are subsequent invocations in the series. Then, in response to encountering a return statement in a source subprogram that is invoked by the last invocation statement in the series, the compiler arrangement generates an instruction to retrieve the stored instruction pointer from the execution stack, and to branch from execution of an object subprogram compiled from the source subprogram that includes the return statement to execution of the object subprogram compiled from the source subprogram that includes the first invocation statement in the series, at the instruction pointed to by the instruction pointer retrieved from the execution stack, and to branch so directly, without in the meantime resuming execution of any object subprogram compiled from a source subprogram that includes an intermediate invocation statement in the series. The compiler arrangement also brings about the additional functionality that has been characterized above, by generating corresponding object instructions.

These and other advantages and features of the present invention will become apparent from the following description of an illustrative embodiment of the invention taken together with the drawing.

BRIEF DESCRIPTION OF THE DRAWING

FIG. **1** is a functional block diagram of a computer that serves as the environment of an illustrative embodiment of the invention;

FIG. **2** is a functional diagram of a conventional CALL-and-RETURN invocation series of the computer of FIG. **1**;

FIGS. **3–5** are flow diagrams of the operations of a conventional compiler in compiling a program that includes the statements of FIG. **2**, and of the execution of the compiled instructions by the computer of FIG. **1**;

FIGS. **6–10** are block diagrams of conventional execution stack manipulations effected by the execution of the compiled instructions of FIGS. **3–5** by the computer of FIG. **1**;

FIG. **11** is a functional diagram of a PASS CONTROL-and-RETURN invocation series of the computer of FIG. **1**;

FIGS. **12–14** are flow diagrams of the operations of a compiler in compiling a program that includes the statements of FIG. **11**, and of the execution of the compiled instructions by the computer of FIG. **1**; and

FIGS. **15–20** are block diagrams of execution stack manipulations effected by the execution of the compiled instructions of FIGS. **12–14** by the computer of FIG. **1**.

DETAILED DESCRIPTION

A simple way to come to understand and appreciate the invention is to contrast it with the conventional call-and-return arrangement. Accordingly, this discussion begins with a rather extensive discussion of that arrangement.

FIG. **1** is a block diagram of an illustrative general purpose data processor operating under stored program control, referred to hereafter as a computer **100**. Computer **100** includes memory **101** and a central processing unit (CPU) **102**. CPU **102** conventionally includes an arithmetic and logic unit (ALU) **606** and a register file **605** including at least an instruction pointer (IP) register **603**, a stack pointer (SP) register **602**, a frame printer (FP) register **601**, and a plurality of general purpose registers **604**. Memory **101** herein represents any form of storage, such as RAM, ROM, and/or disk. It includes a program **111** for execution by CPU **102**, and input information **110** for use by program **111** during its execution. Following execution of program **111** by CPU **102**, memory **100** also includes output information **112** produced by the execution of program **111** while using input information **110**. CPU **102** can therefore be viewed as implementing a functional element **113** defined by the expression

$$f(a)=b$$

wherein f is the function implemented by the executing program **111**,

a is the input information **110**, and

b is the output information **112**.

In order for a program **111** to be executed by computer **100**, the program must be in a form that is understood by logic (e.g. ALU **606**) of computer **100**. This form is commonly referred to as an object program. However, most programs are not initially written in object form, but are written in a high-level, source program form. Consequently, before a program can be executed by computer **100**, it must be convened from source to object form. The conversion is performed by a compiler object program, or compiler for short. Initially, therefore, executing program **111** is a compiler, input information **110** is an application source program, and output information **112** is an application object program created by the compiler from the application source program. Thereafter, the application object program becomes executing program **111**, and it operates on application input information **110** to produce application output information **112**.

During its execution, a program **111** uses an execution stack **114** implemented in memory **101** to temporarily store at least some partial input and output information as well as other temporary or transient information. Briefly turning to FIG. **8**, stack **114** is a last-in, first-out memory structure. Associated with it are two pointers: stack pointer (SP) **602** that points to the next free storage location on stack **114**, and frame pointer (FP) **601** that points to the beginning, the first storage location, of the stack portion that is associated with the presently-executing subprogram. This stack portion extends between the stack pointer and the frame pointer and is referred to as a frame, e.g., frame **610**, **611**, **612**. A focus of this application is the efficient use of stack **114** by the object program that was created by the compiler, and the way in which the compiler brings about that efficient use by the object program.

Application source program **110** may include a chain of subroutine terminal calls and returns such as is shown in FIG. **2**. It should be understood that each subprogram **200** to **209** may include additional calls within its body. These are not considered to be a part of the chain and are not a subject of focus of this application. The application concerns only terminal calls of invoked subprograms, i.e., the last statement performed by an invoked subprogram prior to its returning to the invoking subprogram. As shown in FIG. **2**,

```
File 348:EUROPEAN PATENTS 1978-2005/Sep W03
         (c) 2005 European Patent Office
File 349:PCT FULLTEXT 1979-2005/UB=20050922,UT=20050915
         (c) 2005 WIPO/Univentio

Set     Items   Description
S1        887   FUNCTION? ?(7X)(DERIV??? OR DERIVATION OR INHERIT?)(7X)FUN-
            CTION? ?
S2        168   PROCEDURE? ?(7X)(DERIV??? OR DERIVATION OR INHERIT?)(7X)PR-
            OCEDURE? ?
S3       1434   METHOD? ?(7X)(DERIV??? OR DERIVATION OR INHERIT?)(7X)METHO-
            D? ?
S4      16482   (SHAR??? OR COMMON OR MUTUAL??)(5N)(ATTRIBUTE? ? OR ARGUME-
            NT? ? OR PROPERTY OR PROPERTIES OR PARAMETER? ?)
S5         38   FUNCTION? ?(7X)S4(7X)FUNCTION? ?
S6         10   PROCEDURE? ?(7X)S4(7X)PROCEDURE? ?
S7         44   METHOD? ?(7X)S4(7X)METHOD? ?
S8       2535   S1:S3 OR S5:S7
S9        463   S8 AND IC=G06F
S10       234   S1:S3(50N)(PROGRAM? ? OR PROGRAMMING OR CODE? ? OR CODING -
            OR INSTRUCTIONS)
S11       104   S10 AND IC=G06F
S12       129   S1:S2(50N)(PROGRAM? ? OR PROGRAMMING OR CODE? ? OR CODING -
            OR INSTRUCTIONS)
S13        50   S12 AND IC=G06F
S14        50   IDPAT (sorted in duplicate/non-duplicate order)
S15        22   S5:S7(50N)(PROGRAM? ? OR PROGRAMMING OR CODE? ? OR CODING -
            OR INSTRUCTIONS)
S16        15   S15 AND IC=G06F
S17     15454   SUBROUTINE? ? OR SUBPROGRAM? ? OR SUB()(ROUTINE? ? OR PROG-
            RAM? ?)
S18         7   S17(7X)(DERIV??? OR DERIVATION OR INHERIT? OR S4)(7X)S17
S19        54   S11 NOT S13
```

14/3,K/21 (Item 21 from file: 348)
DIALOG(R)File 348:EUROPEAN PATENTS
(c) 2005 European Patent Office. All rts. reserv.

00752472
Integrating rules into object-oriented programming systems
Integration von Regeln in objektorientierte Programmierungssysteme
L'integration des regles vers des systemes orientes objets
PATENT ASSIGNEE:
AT&T Corp., (589370), 32 Avenue of the Americas, New York, NY 10013-2412, (US), (applicant designated states: DE;FR;GB)
INVENTOR:
Crawford, James Melton, Jr., 29915 Fox Hollow Road, Eugene, Oregon 97405, (US)
Litman, Diane Judith, 9 Everitt Place, Maplewood, New Jersey 07040, (US)
Dvorak, Daniel L., 9 Chestnut Hill Lane, Lincroft, New Jersey 07738, (US)
Mishra, Anil K., 458 Horizon Drive, Edison, New Jersey 08817, (US)
Patel-Schneider, Peter Frederick, 405 Woodland Avenue, Westfield, New Jersey 07090, (US)
LEGAL REPRESENTATIVE:
Buckley, Christopher Simon Thirsk et al (28912), AT&T (UK) LTD., AT&T Intellectual Property Division, 5 Mornington Road, Woodford Green, Essex IG8 0TU, (GB)
PATENT (CC, No, Kind, Date): EP 708401 A2 960424 (Basic)
EP 708401 A3 970319
APPLICATION (CC, No, Date): EP 95307179 951011;
PRIORITY (CC, No, Date): US 327336 941021
DESIGNATED STATES: DE; FR; GB
INTERNATIONAL PATENT CLASS: **G06F-009/44**
ABSTRACT WORD COUNT: 187

LANGUAGE (Publication,Procedural,Application): English; English; English
FULLTEXT AVAILABILITY:

Available Text	Language	Update	Word Count
CLAIMS A	(English)	EPAB96	1004
SPEC A	(English)	EPAB96	14310
Total word count - document A			15314
Total word count - document B			0
Total word count - documents A + B			15314

INTERNATIONAL PATENT CLASS: **G06F-009/44**

...SPECIFICATION from class <class name>(underscore)rpp. The constructor functions are thereby inherited by the class. The constructor **functions** in a preferred embodiment are **inherited** to ensure that the constructor **function** will be executed ahead of any other action associated with creation of an object of the class...
...action associated with destruction of an object of the class.
R++ preprocessor 805 similarly analyzes the source **code** to determine which class data members are mentioned in rule conditions and consequently require modifier functions. Preprocessor...

14/3,K/22 (Item 22 from file: 348)
DIALOG(R)File 348:EUROPEAN PATENTS
(c) 2005 European Patent Office. All rts. reserv.

00738879
Class library for use in graphics programming
Klassenbibliothek fur die graphische Programmierung
Bibliotheque de classe a utiliser pour programmation graphique
PATENT ASSIGNEE:
AT&T Corp., (589370), 32 Avenue of the Americas, New York, NY 10013-2412, (US), (Proprietor designated states: all)
INVENTOR:

Eick, Stephen Gregory, 1413 Durness Court, Naperville, Illinois 60565, (US)
Wills, Graham John, 2733 Buckingham Drive, Apt. 104, Lisle, Illinois 60532, (US)
Lucas, Paul Jay, 18 Huntington Circle West, Apt. 10, Naperville, Illinois 60540, (US)
LEGAL REPRESENTATIVE:
Watts, Christopher Malcolm Kelway, Dr. et al (37391), Lucent Technologies (UK) Ltd, 5 Mornington Road, Woodford Green Essex, IG8 0TU, (GB)
PATENT (CC, No, Kind, Date): EP 696770 A1 960214 (Basic)
EP 696770 B1 020828
APPLICATION (CC, No, Date): EP 95303903 950607;
PRIORITY (CC, No, Date): US 260133 940615
DESIGNATED STATES: DE; FR; GB
INTERNATIONAL PATENT CLASS: **G06F-009/44**
ABSTRACT WORD COUNT: 156
NOTE:
Figure number on first page: 2

LANGUAGE (Publication,Procedural,Application): English; English; English
FULLTEXT AVAILABILITY:

Available Text	Language	Update	Word Count
CLAIMS A	(English)	EPAB96	488
CLAIMS B	(English)	200235	596
CLAIMS B	(German)	200235	545
CLAIMS B	(French)	200235	689
SPEC A	(English)	EPAB96	12056
SPEC B	(English)	200235	11851
Total word count - document A			12546
Total word count - document B			13681
Total word count - documents A + B			26227

INTERNATIONAL PATENT CLASS: **G06F-009/44**

...SPECIFICATION function specifications 104. An ordinary function specification 102 specifies an interface for the operation and also contains **code** for doing the operation. A virtual function specification only specifies the interface. **Code** for doing the operation must be provided in a separate ordinary **function** 104, and different class specifications 103 which **inherit** a class having a virtual **function** may have different ordinary functions 104 for the virtual function. The only requirement is that all of...

...SPECIFICATION function specifications 104. An ordinary function specification 104 specifies an interface for the operation and also contains **code** for doing the operation. A virtual function specification only specifies the interface. **Code** for doing the operation must be provided in a separate ordinary **function** 104, and different class specifications 103 which **inherit** a class having a virtual **function** may have different ordinary functions 104 for the virtual function. The only requirement is that all of...

14/3,K/24 (Item 24 from file: 348)
DIALOG(R)File 348:EUROPEAN PATENTS

00675866
Information catalog system with object-dependent functionality
Informationsarchivierungssystem mit objektabhangiger Funktionalitat
Systeme d'archivage d'informations avec une fonctionnalite dependant de l'objet
PATENT ASSIGNEE:
International Business Machines Corporation, (200120), Old Orchard Road, Armonk, N.Y. 10504, (US), (Proprietor designated states: all)

INVENTOR:
  Harper, Lloyd, 7144 Via Romera, San Jose, California 95139, (US)
  Labrie, Jacques, 1415 Hervey Lane, San Jose, California 95125, (US)
LEGAL REPRESENTATIVE:
  Burt, Roger James, Dr. (52152), IBM United Kingdom Limited Intellectual Property Department Hursley Park, Winchester Hampshire SO21 2JN, (GB)
PATENT (CC, No, Kind, Date): EP 647909 A1 950412 (Basic)
                             EP 647909 B1 030416
APPLICATION (CC, No, Date): EP 94306033 940816;
PRIORITY (CC, No, Date): US 134355 931008
DESIGNATED STATES: DE; FR; GB
INTERNATIONAL PATENT CLASS: **G06F-017/30**
ABSTRACT WORD COUNT: 170
NOTE:
  Figure number on first page: 1

LANGUAGE (Publication,Procedural,Application): English; English; English
FULLTEXT AVAILABILITY:

Available Text	Language	Update	Word Count
CLAIMS A	(English)	EPAB95	383
CLAIMS B	(English)	200316	492
CLAIMS B	(German)	200316	477
CLAIMS B	(French)	200316	647
SPEC A	(English)	EPAB95	5075
SPEC B	(English)	200316	5126
Total word count - document A			5458
Total word count - document B			6742
Total word count - documents A + B			12200

INTERNATIONAL PATENT CLASS: **G06F-017/30**

...SPECIFICATION object instance identified. In this way, the meta objects of the database catalog system 26 encapsulate the **functions** and properties which they **inherit** from the **function** category classes to which they belong. Such encapsulation could be similarly achieved using object oriented **programming** tools such as the C++ **programming** language.
  META OBJECT DATA STRUCTURES
    Referring now to Fig. 4, an exemplary structure of a meta data...

...SPECIFICATION instance identified. In this way, the user-defined objects of the database catalog system 26 encapsulate the **functions** and properties which they **inherit** from the **function** category classes to which they belong. Such encapsulation could be similarly achieved using object oriented **programming** tools such as the C++ **programming** language.

  USER-DEFINED OBJECT DATA STRUCTURES
    Referring now to Fig. 4, an exemplary structure of a user...

14/3,K/25 (Item 25 from file: 348)
DIALOG(R)File 348:EUROPEAN PATENTS

00638555
Object system with derived metaclasses.
Objektsystem mit abgeleiteten Metaklassen.
Systeme objet avec metaclasses derivees.
PATENT ASSIGNEE:
  International Business Machines Corporation, (200120), Old Orchard Road, Armonk, N.Y. 10504, (US), (applicant designated states: DE;FR;GB)
INVENTOR:
  Danforth, Scott Harrison, 10011 Woodland Village Drive, Austin, Texas 78750, (US)

LEGAL REPRESENTATIVE:
Burt, Roger James, Dr. (52152), IBM United Kingdom Limited Intellectual Property Department Hursley Park, Winchester Hampshire SO21 2JN, (GB)
PATENT (CC, No, Kind, Date): EP 619544 A2 941012 (Basic)
EP 619544 A3 950201
APPLICATION (CC, No, Date): EP 94301931 940317;
PRIORITY (CC, No, Date): US 42930 930405
DESIGNATED STATES: DE; FR; GB
INTERNATIONAL PATENT CLASS: **G06F-009/44**
ABSTRACT WORD COUNT: 141
LANGUAGE (Publication,Procedural,Application): English; English; English
FULLTEXT AVAILABILITY:

Available Text	Language	Update	Word Count
CLAIMS A	(English)	EPABF2	472
SPEC A	(English)	EPABF2	15716
Total word count - document A			16188
Total word count - document B			0
Total word count - documents A + B			16188

INTERNATIONAL PATENT CLASS: **G06F-009/44**

...SPECIFICATION are of the same class as this object also contain an address that points to this method **procedure** table diagrammed at label 240. Any methods **inherited** by the objects will have their method **procedure** addresses at the same offset in memory as they appear in the method procedure table as set...

...class from which it is inherited.
Addresses of the blocks of computer memory containing the series of **instructions** for two of the method procedures are set forth at labels 250 and 260. Labels 270 and...are of the same class as this object also contain an address that points to this method **procedure** table diagrammed at label 1240. Any methods **inherited** by the objects will have their method **procedure** addresses at the same offset in memory as they appear in the method procedure table as set 1270. A redispatch stub is a sequence of **instructions** that appear as a method to a client program. However, the instructions merely convert the method call...

14/3,K/28 (Item 28 from file: 348)
DIALOG(R)File 348:EUROPEAN PATENTS

00546477
Computer with object oriented environment.
Rechner mit einer objektorientierten Umgebung.
Ordinateur avec un environnement oriente objet.
PATENT ASSIGNEE:
International Business Machines Corporation, (200120), Old Orchard Road, Armonk, N.Y. 10504, (US), (applicant designated states: CH;DE;FR;GB;IT;LI;NL;SE)
INVENTOR:
Conner, Mike H., 4416 Walhill Lane, Austin, Texas 78759, (US)
Martin, Andrew R., 1070 Mearns Meadow No. 534, Austin, Texas 78758, (US)
Raper, Larry K., 7860 Lakewood Drive, Austin, Texas 78750, (US)
LEGAL REPRESENTATIVE:
Burt, Roger James, Dr. (52152), IBM United Kingdom Limited Intellectual Property Department Hursley Park, Winchester Hampshire SO21 2JN, (GB)
PATENT (CC, No, Kind, Date): EP 546809 A2 930616 (Basic)
EP 546809 A3 931118
APPLICATION (CC, No, Date): EP 92311207 921209;
PRIORITY (CC, No, Date): US 805779 911212
DESIGNATED STATES: CH; DE; FR; GB; IT; LI; NL; SE
INTERNATIONAL PATENT CLASS: **G06F-009/44**
ABSTRACT WORD COUNT: 151

LANGUAGE (Publication,Procedural,Application): English; English; English
FULLTEXT AVAILABILITY:

Available Text	Language	Update	Word Count
CLAIMS A	(English)	EPABF1	320
SPEC A	(English)	EPABF1	14070
Total word count - document A			14390
Total word count - document B			0
Total word count - documents A + B			14390

INTERNATIONAL PATENT CLASS: **G06F-009/44**

...SPECIFICATION are of the same class as this object also contain an address that points to this method **procedure** table diagrammed at label 240. Any methods **inherited** by the objects will have their method **procedure** addresses at the same offset in memory as they appear in the method procedure table as set...class from which it is inherited.
Addresses of the blocks of computer memory containing the series of **instructions** for two of the method procedures are set forth at labels 250 and 260. Labels 270 and...are of the same class as this object also contain an address that points to this method **procedure** table diagrammed at label 1240. Any methods **inherited** by the objects will have their method **procedure** addresses at the same offset in memory as they appear in the method procedure table as set...

...label 1250 contains a pointer to a redispatch stub 1270. A redispatch stub is a sequence of **instructions** that appear as a method to a client program. However, the instructions merely convert the method call...

14/3,K/30 (Item 30 from file: 348)
DIALOG(R)File 348:EUROPEAN PATENTS

00545574
Language neutral objects
Sprachenneutrale Objekte
Objets neutres vis-a-vis du langage
PATENT ASSIGNEE:
International Business Machines Corporation, (200120), Old Orchard Road, Armonk, N.Y. 10504, (US), (Proprietor designated states: all)
INVENTOR:
Conner, Mike H., 4416 Walhill Lane, Austin, Texas 78759, (US)
Martin, Andrew R., 1070 Mearns Meadow No.534, Austin, Texas 78758, (US)
Raper, Larry K., 7860 Lakewood Drive, Austin, Texas 78750, (US)
LEGAL REPRESENTATIVE:
Burt, Roger James, Dr. (52152), IBM United Kingdom Limited Intellectual Property Department Hursley Park, Winchester Hampshire SO21 2JN, (GB)
PATENT (CC, No, Kind, Date): EP 546683 A2 930616 (Basic)
EP 546683 A3 931201
EP 546683 B1 000119
APPLICATION (CC, No, Date): EP 92310241 921109;
PRIORITY (CC, No, Date): US 805668 911212
DESIGNATED STATES: CH; DE; FR; GB; IT; LI; NL; SE
INTERNATIONAL PATENT CLASS: **G06F-009/44** ; **G06F-009/45**
ABSTRACT WORD COUNT: 108
NOTE:
Figure number on first page: 4

LANGUAGE (Publication,Procedural,Application): English; English; English
FULLTEXT AVAILABILITY:

Available Text	Language	Update	Word Count
CLAIMS B	(English)	200003	99
CLAIMS B	(German)	200003	97
CLAIMS B	(French)	200003	126

SPEC B (English) 200003 13127
Total word count - document A 0
Total word count - document B 13449
Total word count - documents A + B 13449

INTERNATIONAL PATENT CLASS: **G06F-009/44** ...

... **G06F-009/45**

...SPECIFICATION are of the same class as this object also contain an address that points to this method **procedure** table diagrammed at label 240. Any methods **inherited** by the objects will have their method **procedure** addresses at the same offset in memory as they appear in the method procedure table as set...

...class from which it is inherited.
Addresses of the blocks of computer memory containing the series of **instructions** for two of the method procedures are set forth at labels 250 and 260. Labels 270 and...are of the same class as this object also contain an address that points to this method **procedure** table diagrammed at label 1240. Any methods **inherited** by the objects will have their method **procedure** addresses at the same offset in memory as they appear in the method procedure table as set...

...label 1250 contains a pointer to a redispatch stub 1270. A redispatch stub is a sequence of **instructions** that appear as a method to a client program. However, the instructions merely convert the method call...

14/3,K/34 (Item 34 from file: 348)
DIALOG(R)File 348:EUROPEAN PATENTS

00254150
Apl-to-fortran translator
APL-Fortran-Ubersetzer
Traducteur APL-fortran
PATENT ASSIGNEE:
International Business Machines Corporation, (200120), Old Orchard Road, Armonk, N.Y. 10504, (US), (applicant designated states: DE;FR;GB;IT)
INVENTOR:
Driscoll, Graham Cameron, 9 Charlotte Place, Hartsdale New York 10530, (US)
Orth, Donald Lawrence, Cherry Hill Road RFD1, Carmel New York 10512, (US)
LEGAL REPRESENTATIVE:
Schafer, Wolfgang, Dipl.-Ing. et al (62021), IBM Deutschland Informationssysteme GmbH Patentwesen und Urheberrecht, D-70548 Stuttgart, (DE)
PATENT (CC, No, Kind, Date): EP 252229 A2 880113 (Basic)
EP 252229 A3 920311
EP 252229 B1 960626
APPLICATION (CC, No, Date): EP 87105698 870416;
PRIORITY (CC, No, Date): US 882737 860707
DESIGNATED STATES: DE; FR; GB; IT
INTERNATIONAL PATENT CLASS: **G06F-009/44**
ABSTRACT WORD COUNT: 171

LANGUAGE (Publication,Procedural,Application): English; English; English
FULLTEXT AVAILABILITY:
Available Text Language Update Word Count
CLAIMS B (English) EPAB96 2317
CLAIMS B (German) EPAB96 2222
CLAIMS B (French) EPAB96 2556
SPEC B (English) EPAB96 17757
Total word count - document A 0

Total word count - document B 24852
Total word count - documents A + B 24852

INTERNATIONAL PATENT CLASS: **G06F-009/44**

...SPECIFICATION a larger array, the shape corresponds to the number of elements in the respective dimensions.

In APL **programming** , primitive **functions** and **derived functions** may be used by themselves to perform desired operations or may serve as building blocks that can...

...When a user defines a function, it may be "called" in the same way that a primitive **function** or **derived function** is. That is, by specifying arguments (as required) and including an appropriate reference to the function, the...prior technology fails to adequately account for the various cases which can apply to the APL primitive **functions** and **derived functions** or fail to overcome the difficulty of translating variables which the language does not distinguish based on...

...deriving shape information --which severely limits translation or compilation.

The following U.S.Patents relate to computer **program** translation in general. U.S. Patent 4374408 discloses a multi-pass system for translating an RPG (report...some point compiled.

Referring now to Figure 3, a translator 200 includes several major elements. Source language **code** is re-structured, for each **program** , into a sequence of simple APL expressions which are stored in a master table 202. A simple expression may correspond to a primitive function (e.g. a **function** or an operator), a **derived function** (such as the reduction **functions**), an idiom (discussed hereinbelow), or a user-defined function. Each of these functions may be typically characterized...

...discussed hereinbelow, which also aids in facilitating the translating procedure is also derived.

The re-structured APL **code** and the information (and attributes) derived therefor are applied to stored archetypes 206 by a target language generator 208. Each stored archetype includes **code** representative of target language **code** associated with a corresponding primitive **function** or **derived function** . For each particular primitive **function** or **derived function** occurring in the master table, there is thus a corresponding archetype. To account for the various cases applicable to each primitive **function** or **derived function** , each archetype includes (a) appropriate conditions and (b) portions of **code** which can be selectively generated for each case based on the resolution of conditions. Based on information derived by the static analyzer 204 or supplied by the user through declarations, the target language **code** generator 208 selects, or generates, appropriate target language code from the archetypes. The target language code generator...

...next program is processed beginning with a syntax analysis in step 304. The syntax analysis transforms the **program** being processed into a sequence of simple APL expressions of the form: (Table omitted)

For example, a would be stored as: (Table omitted)

The term **function** refers to primitive APL **function** , **derived function** , or user-defined **function** . If the **function** is a user-defined APL function and that name is not on the list of **programs** to be translated, the name thereof is appended to the **program** list.

In step 306, a branch expression analysis is performed. In order to avoid potential degradation in analyzing APL **programs** , branch expressions are limited to those whose branch targets can be determined during syntax analysis. Branch targets...

...value are noted and included in classes of equivalents (step 406). The formal definitions of APL primitive **functions** and **derived function** are used to determine identical variables. For each equivalent class, a

representative variable is selected --preferably the variable in the class which occurs first in the **program** execution (step 408). Thereafter, a simple expression is built for each representative in step 410. These simple...

14/3,K/40 (Item 40 from file: 349)
DIALOG(R)File 349:PCT FULLTEXT
(c) 2005 WIPO/Univentio. All rts. reserv.

01012892 **Image available**
METHOD AND SYSTEM FOR SOFTWARE APPLICATION DEVELOPMENT AND CUSTOMIZABLE RUNTIME ENVIRONMENT
PROCEDE ET SYSTEME DESTINES AU DEVELOPPEMENT D'UNE APPLICATION LOGICIELLE ET A UN ENVIRONNEMENT D'EXECUTION PERSONNALISABLE
Patent Applicant/Assignee:
EXEGESYS INC, 144 South 500 East, Salt Lake City, Utah 84102, US, US (Residence), US (Nationality)
Inventor(s):
SEAMAN Christopher G, 9727 S. Skye Park Road, South Jordan, Utah 84095, US,
JAMES Bruce Alan, 1555 E. Parkridge Drive, Salt Lake City, UT 84121, US,
QUINN Steven J, 659 West Halifax Court, Farmington, UT 84025, US,
DORIUS Paul F, 8870 North Silver Spur Road, Park City, UT 84098, US,
PETERSEN Scott T, 1238 East Eagle Way, Fruit Heights, UT 84037, Utah 84037, US,
MILLER Keven D, 484 E. 700 North, Orem, UT 84097, US,
Legal Representative:
SADLER Lloyd W (et al) (agent), Parsons, Behle & Latimer, 201 South Main Street, Suite 1800, Salt Lake City, UT 84111, US,
Patent and Priority Information (Country, Number, Date):
Patent: WO 200342823 A1 20030522 (WO 0342823)
Application: WO 2002US36984 20021114 (PCT/WO US0236984)
Priority Application: US 2001332345 20011114
Designated States:
(Protection type is "patent" unless otherwise stated - for applications prior to 2004)
AE AG AL AM AT AU AZ BA BB BG BR BY BZ CA CH CN CO CR CU CZ DE DK DM DZ
EC EE ES FI GB GD GE GH GM HR HU ID IL IN IS JP KE KG KP KR KZ LC LK LR
LS LT LU LV MA MD MG MK MN MW MX MZ NO NZ OM PH PL PT RO RU SD SE SG SI
SK SL TJ TM TN TR TT TZ UA UG UZ VN YU ZA ZM ZW
(EP) AT BE BG CH CY CZ DE DK EE ES FI FR GB GR IE IT LU MC NL PT SE SK TR
(OA) BF BJ CF CG CI CM GA GN GQ GW ML MR NE SN TD TG
(AP) GH GM KE LS MW MZ SD SL SZ TZ UG ZM ZW
(EA) AM AZ BY KG KZ MD RU TJ TM
Publication Language: English
Filing Language: English
Fulltext Word Count: 15653

Main International Patent Class: **G06F-009/44**
Fulltext Availability:
Detailed Description
Detailed Description
... present preferred generic data transfer functions 502 base class is the XCTransport class, contained in the source **code** files transport.h and transport.cpp. Pipe **functions** 503 represents a class **derived** from XCTransport that contains data transfer **functions** specifically for OS pipes. The derived class in the present embodiment of the invention is XCPipeTransport class, contained in the source **code** files pipes.h and pipes.cpp. The File Manipulation **Functions** 504 represents a class **derived** from XCTransport that contains data transfer **functions** and manipulation functions adapted specifically for OS functions. The derived class is the XCHIeTransport class, contained in the source **code** files files.h and files.cpp.

Network Socket **Functions** 505 represents a class **derived** from XCTransport that contains data transfer **functions** specifically for the OS Network Sockets. The derived class is the XCSocketTransport class, contained in the source **code** files socket.h and socket.cpp. The Service Functions 506 in the present embodiment represents functions for...

14/3,K/44 (Item 44 from file: 349)
DIALOG(R)File 349:PCT FULLTEXT

00753772 **Image available**
METHOD, SYSTEM AND COMPUTER PROGRAM PRODUCT FOR NON-LINEAR MAPPING OF MULTI-DIMENSIONAL DATA
PROCEDE, SYSTEME ET PROGRAMME INFORMATIQUE D'APPLICATION NON LINEAIRE DE DONNEES MULTIDIMENSIONNELLES
Patent Applicant/Assignee:
3-DIMENSIONAL PHARMACEUTICALS INC, Eagleview Corporate Center, Suite 104, 665 Stockton Drive, Exton, PA 19341, US, US (Residence), US (Nationality)
Inventor(s):
AGRAFIOTIS Dimitris K, 660 Perimeter Drive, Downingtown, PA 19335, US
LOBANOV Victor S, 24305 Cornerstone Drive, Yardley, PA 19067, US
SALEMME Francis R, 1970 Timber Lakes, Yardley, PA 19067, US
Legal Representative:
LEE Michael Q, Sterne, Kessler, Goldstein & Fox P.L.L.C., Suite 600, 1100 New York Avenue, N.W., Washington, DC 20005-3934, US
Patent and Priority Information (Country, Number, Date):
Patent: WO 200067148 A1 20001109 (WO 0067148)
Application: WO 2000US11838 20000503 (PCT/WO US0011838)
Priority Application: US 99303671 19990503
Designated States:
(Protection type is "patent" unless otherwise stated - for applications prior to 2004)
AE AG AL AM AT AU AZ BA BB BG BR BY CA CH CN CR CU CZ DE DK DM DZ EE ES FI GB GD GE GH GM HR HU ID IL IN IS JP KE KG KP KR KZ LC LK LR LS LT LU LV MA MD MG MK MN MW MX NO NZ PL PT RO RU SD SE SG SI SK SL TJ TM TR TT TZ UA UG UZ VN YU ZA ZW
(EP) AT BE CH CY DE DK ES FI FR GB GR IE IT LU MC NL PT SE
(OA) BF BJ CF CG CI CM GA GN GW ML MR NE SN TD TG
(AP) GH GM KE LS MW SD SL SZ TZ UG ZW
(EA) AM AZ BY KG KZ MD RU TJ TM
Publication Language: English
Filing Language: English
Fulltext Word Count: 14837

Main International Patent Class: **G06F-017/17**
Fulltext Availability:
Detailed Description

Detailed Description
... tables, software modules and/or sub-routines, hardware, and combinations thereof.
In an embodiment, the non-linear **function** (s) determining module 1518 determines, or **derives** , the one or more non-linear **functions** using any of a variety of techniques including self learning or organizing techniques such as, but not...

...optimization techniques including, but not limited to, Monte Carlo/random sampling, greedy search algorithms, simulated annealing, evolutionary **programming** , genetic algorithms, genetic **programming** , gradient minimization techniques, and combinations thereof
The one or more non-linear functions 1519 are provided to...

14/3,K/47 (Item 47 from file: 349)
DIALOG(R)File 349:PCT FULLTEXT

00327722
OBJECT ORIENTED DATA STORE INTEGRATION ENVIRONMENT
ENVIRONNEMENT D'INTEGRATION DE MEMOIRE DE DONNEES ORIENTEES-OBJET
Patent Applicant/Assignee:
ONTOS INC,
Inventor(s):
MARTEL Paul A,
HARRIS Craig S,
Patent and Priority Information (Country, Number, Date):
Patent: WO 9610232 A1 19960404
Application: WO 95US2549 19950302 (PCT/WO US9502549)
Priority Application: US 94315394 19940929
Designated States:
(Protection type is "patent" unless otherwise stated - for applications prior to 2004)
JP AT BE CH DE DK ES FR GB GR IE IT LU MC NL PT SE
Publication Language: English
Fulltext Word Count: 26706

Main International Patent Class: **G06F-017/30**
Fulltext Availability:
Detailed Description

Detailed Description
... Many OC.ExternalSM functions translate an invocation of an OC-ExternalSM function into an invocation of a **function** on an OC.ExternalKey- or OC.ExtemalTypeMap- **derived** class. This delegation of **functions** to component classes supports **code** reuse, as a
single OC.ExternaISM subclass may use several instances of a single component class or...

14/3,K/50 (Item 50 from file: 349)
DIALOG(R)File 349:PCT FULLTEXT

00117801
METHOD AND APPARATUS FOR HANDLING INTERPROCESSOR CALLS IN A MULTIPROCESSOR SYSTEM
PROCEDE ET APPAREIL PERMETTANT D'ASSURER L'ECOULEMENT DU TRAFIC D'APPEL ENTRE PROCESSEURS DANS UN SYSTEME A MULTIPROCESSEURS
Patent Applicant/Assignee:
WESTERN ELECTRIC COMPANY INC,
Inventor(s):
DeBRULER Dennis L,
Patent and Priority Information (Country, Number, Date):
Patent: WO 8401043 A1 19840315
Application: WO 83US435 19830328 (PCT/WO US8300435)
Priority Application: US 82899 19820826
Designated States:
(Protection type is "patent" unless otherwise stated - for applications prior to 2004)
AT BE CH DE FR GB JP LU NL SE
Publication Language: English
Fulltext Word Count: 6662

Main International Patent Class: **G06F-015/16**
International Patent Class: **G06F-09:46**
Fulltext Availability:

Detailed Description

Detailed Description
... execute the next
requested program function.

In the case of a function call in which the calling **program** function expects a return, the return from the called **program** function is implemented as a call to the original calling function. The return address data is stored in the **program** function context by the original calling **program** function and found there by the called OMPI
program **function** ; values **derived** by the called **program** **function** are stored in locations specified by the calling **program** function4
In this embodiment, **program** function context, output queues, work queues, and their associated pointers are all in shared memory. Since facilities...

19/9/17 (Item 17 from file: 348)
DIALOG(R)File 348:EUROPEAN PATENTS

00856191
METHOD AND SYSTEM FOR IMPLEMENTING SOFTWARE OBJECTS
VERFAHREN UND VORRICHTUNG ZUR EINFUHRUNG VON SOFTWARE-OBJEKTEN
PROCEDE ET DISPOSITIF DE MISE EN OEUVRE D'OBJETS LOGICIELS
PATENT ASSIGNEE:
Design Intelligence, Inc., (2234430), Suite 2650, 1111 Third Avenue, Seattle, WA 98101, (US), (Proprietor designated states: all)
INVENTOR:
McDONALD, Marc, B., 6146 West Mercer Way, Mercer Island, WA 98040, (US)
ORR, Michael, B., 7009 N.E. Dolphin Drive, Bainbridge Island, WA 98110, (US)
LEGAL REPRESENTATIVE:
Hoarton, Lloyd Douglas Charles et al (80191), Forrester & Boehmert, Franz-Joseph-Strasse 38, 80801 Munchen, (DE)
PATENT (CC, No, Kind, Date): EP 856170 A1 980805 (Basic)
EP 856170 B1 000105
WO 9715005 970424
APPLICATION (CC, No, Date): EP 96936557 961016; WO 96US16560 961016
PRIORITY (CC, No, Date): US 546316 951020

Another approach to object-oriented language architecture is the use of a containment, or "has-a," hierarchy as opposed to an is-a hierarchy. In a containment hierarchy, a child object is contained in a parent object, instead of being a type of the parent object. An object is contained within only a single container, although the parent container can itself be contained within another container.

Most implementations do not allow inheritance from the parent in a container hierarchy. In some applications, such as Hypercard(TM) and Toolbook(TM), when a method cannot be found associated with an object, the container of the object is checked to determine whether the container defines the **method** . Such a technique can be considered **inheritance** of **methods** from a parent container.

In summary, the class-instance **programming** model uses abstract classes that define a structure and methods. The structure and methods of one or more abstract classes can be inherited by a child abstract class. Inheritance in the class-instance model is based upon an is-a hierarchy, in which child classes are a type of the parent class and a child class inherits everything in its parent class or classes. Instances, which include property values, are created from the abstract classes, but instances cannot be derived from other instances. Therefore, structure and methods can be inherited, but property values cannot be. This restricts reusability and makes the programmer's task more complex. Expert knowledge of complex rules of arbitration and of the parent classes are often required to create specific child classes and to understand inherited behavior.

```
File 347:JAPIO Nov 1976-2005/Apr(Updated 050801)
         (c) 2005 JPO & JAPIO
File 350:Derwent WPIX 1963-2005/UD,UM &UP=200561
         (c) 2005  Thomson Derwent

Set     Items   Description
S1    5429087   FUNCTION? ? OR PROCEDURE? ? OR METHOD? ?
S2       4409   SUBROUTINE? ? OR SUBPROGRAM? ? OR SUB()(ROUTINE? ? OR PROG-
            RAM? ?)
S3    3806862   DERIV? OR BASE? ? OR BASING OR BASIS
S4    1158392   ATTRIBUTE? ? OR ARGUMENT? ? OR PROPERTY OR PROPERTIES OR P-
            ARAMETER? ?
S5     731263   SHAR??? OR COMMON OR MUTUAL??
S6       2349   INHERIT?
S7       4650   S1(7X)S3(7X)S1
S8       1403   S7 AND IC=G06F
S9        489   S8 AND (PROGRAM? ? OR PROGRAMMING OR CODE? ? OR CODING)
S10       418   S1(7X)(DERIV??? OR DERIVATION)(7X)S1
S11        84   S10 AND (PROGRAM? ? OR PROGRAMMING OR CODE? ? OR CODING)
S12        37   S11 AND IC=G06F
S13        11   S12 AND AC=US/PR AND AY=(1970:2001)/PR
S14        12   S12 AND AC=US AND AY=1970:2001
S15        12   S12 AND AC=US AND AY=(1970:2001)/PR
S16        10   S12 AND PY=1970:2001
S17        18   S13:S16
S18      1476   FUNCTION? ?(7X)(BASE? ? OR BASING OR BASIS)(7X)FUNCTION? ?
S19       340   S18 AND (PROGRAM? ? OR PROGRAMMING OR CODE? ? OR CODING)
S20       215   S19 AND IC=G06F
S21         0   S20 AND INHERIT?
S22      4198   S5(5N)S4
S23         0   S20 AND S22
S24        28   S2(7X)S3(7X)S2
S25        12   S1:S2(7X)S22(7X)S1:S2
S26        17   S1(7X)S6(7X)S1
S27        15   S10 AND INSTRUCTIONS
S28         6   S27 AND IC=G06F
S29         1   S28 NOT S12
S30        40   S18 AND INSTRUCTIONS
S31        27   S30 AND IC=G06F
S32         0   S31 AND INHERIT?
```

17/5/2 (Item 2 from file: 347)
DIALOG(R)File 347:JAPIO

04385963 **Image available**
ALGEBRAIC CURVE **CODE** DECODING SYSTEM

PUB. NO.: 06-029863 [JP 6029863 A]
PUBLISHED: February 04, 1994 (**19940204**)
INVENTOR(s): MIURA SHINJI
APPLICANT(s): NEC CORP [000423] (A Japanese Company or Corporation), JP (Japan)
APPL. NO.: 03-296874 [JP 91296874]
FILED: November 13, 1991 (19911113)
INTL CLASS: [5] H03M-013/00; **G06F-011/10**
JAPIO CLASS: 42.4 (ELECTRONICS -- Basic Circuits); 45.1 (INFORMATION PROCESSING -- Arithmetic Sequence Units)
JOURNAL: Section: E, Section No. 1545, Vol. 18, No. 240, Pg. 89, May 09, 1994 (19940509)

ABSTRACT
PURPOSE: To obtain an algebraic curve **code** decoding system with a higher correction capability by further generally correcting and expanding a basic decoding algorithm and a correction decoding algorithm.
CONSTITUTION: When an error location is estimated from the block type data train for which an error correction encoding is performed by the **codes** using algebraic curves, plural effective factors designating the range that can be taken on by the poles of a rational function. The effective factors are arranged in the ascending order of the degree, simultaneous linear equations having the components of syndrome arrays adjoint to the effective factors and the block type data train are successively solved by a procedure A6. By a procedure A7, the solution is stopped at the point where the simultaneous linear equations have the first nonzero solution, the rational **function** is **derived** from the solution by a **procedure** A8, the zero point of the rational function is derived and the location corresponding to the zero point is estimated as an error location by a procedure A9.

17/5/3 (Item 3 from file: 347)
DIALOG(R)File 347:JAPIO

02181533 **Image available**
INFORMATION PROCESSOR

PUB. NO.: 62-098433 [JP 62098433 A]
PUBLISHED: May 07, 1987 (**19870507**)
INVENTOR(s): SUZUKI NORIHISA
NOJIRI TORU
NAKANO KOICHI
KAWASAKI SHUNPEI
APPLICANT(s): HITACHI LTD [000510] (A Japanese Company or Corporation), JP (Japan)
APPL. NO.: 60-237298 [JP 85237298]
FILED: October 25, 1985 (19851025)
INTL CLASS: [4] **G06F-009/44** ; **G06F-007/28**
JAPIO CLASS: 45.1 (INFORMATION PROCESSING -- Arithmetic Sequence Units); 45.2 (INFORMATION PROCESSING -- Memory Units)
JAPIO KEYWORD: R131 (INFORMATION PROCESSING -- Microcomputers & Microprocessers)
JOURNAL: Section: P, Section No. 624, Vol. 11, No. 308, Pg. 28, October 08, 1987 (19871008)

ABSTRACT
PURPOSE: To improve a **program** executing speed by providing a field for

containing the attribute of a method, such as a physical address, etc. of an executing **code** of a method to be called, in a method cache, so that when the method which is called once is called again, the executing **code** can be read out directly and executed.

CONSTITUTION: In consideration of the locality of a **program** , as for a method which is called once, an attribute of its method is contained instead of a method handle and a primitive index, in the third and the fourth areas of each field in a method cache MC. In such a case, a the attribute of the method signifies, for instance, a physical address instruction address for showing an address in which a **code** of an instruction to be executed in the next time is contained, and a physical address (head address) for indicating the address of head part of the method. As for the method which is called once, a physical address for indicating its executing **code** is contained in the method cache MC, therefore, in case of searching its **method** again, the **method** to be **derived** can be called immediately by subtracting the **method** cash MC by a hash value.

17/5/4 (Item 1 from file: 350)
DIALOG(R)File 350:Derwent WPIX

017180310 **Image available**
WPI Acc No: 2005-503927/200551
XRPX Acc No: N05-411061
Compressible fluid simulating method , **involves** deriving **advected distribution** function **from evaluated distribution** function , **and calculating current value of variable from advected distribution function**
Patent Assignee: GEN ATOMICS (GEAT)
Inventor: HINTON F L; ROSENBLUTH M N
Number of Countries: 001 Number of Patents: 001
Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week	
US 6915245	B1	20050705	US 2000233758	P	20000914	200551	B
			US 2001953573	A	20010914		

Priority Applications (No Type Date): US 2000233758 P 20000914; US 2001953573 A 20010914
Patent Details:

Patent No	Kind	Lan	Pg	Main IPC	Filing Notes
US 6915245	B1		15	G06F-019/20	Provisional application US 2000233758

Abstract (Basic): US 6915245 B1
NOVELTY - The method involves selecting an equilibrium distribution function for simulating a fluid. Another distribution function is evaluated and set equal to the former distribution **function** . An advected distribution **function** is **derived** from the evaluated distribution **function** . A current value of a variable is calculated from the advected distribution function. The current value of the variable is generated as output.
DETAILED DESCRIPTION - An INDEPENDENT CLAIM is also included for a computer **program** product having a computer **program** with instructions to perform a method of simulating a compressible fluid.
USE - Used for simulating a compressible fluid.
ADVANTAGE - The method effectively simulates the compressible fluid to provide inherent positive hyperviscosity and hyperdiffusivity for very small values of viscosity and thermal diffusivity.
DESCRIPTION OF DRAWING(S) - The drawing shows a flowchart of a method of simulating a compressible fluid.
pp; 15 DwgNo 1/4
Title Terms: COMPRESS; FLUID; SIMULATE; METHOD; DERIVATIVE; DISTRIBUTE; FUNCTION; EVALUATE; DISTRIBUTE; FUNCTION; CALCULATE; CURRENT; VALUE; VARIABLE; DISTRIBUTE; FUNCTION

Derwent Class: T01
International Patent Class (Main): **G06F-019/20**
File Segment: EPI

17/5/5 (Item 2 from file: 350)
DIALOG(R)File 350:Derwent WPIX

016653855 **Image available**
WPI Acc No: 2004-812575/200480
Related WPI Acc No: 2003-060947; 2003-606685; 2003-688237; 2003-756154; 2003-802072; 2004-430451; 2004-430524; 2005-180482; 2005-512089
XRPX Acc No: N04-641158
Transformation function deriving method **involves mapping source schema and target schema to common ontology model for transforming data conforming to source schema into data conforming to target schema**
Patent Assignee: HELLMAN Z Z (HELL-I); SCHREIBER M Z (SCHR-I); YUVAL T Y (YUVA-I)
Inventor: HELLMAN Z Z; SCHREIBER M Z; YUVAL T Y
Number of Countries: 001 Number of Patents: 001
Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week
US 20040216030	A1	20041028	US 2001866101	A	20010525	200480 B
			US 200253045	A	20020115	

Priority Applications (No Type Date): US 200253045 A 20020115; US 2001866101 A 20010525
Patent Details:

Patent No	Kind	Lan	Pg	Main IPC	Filing Notes
US 20040216030	A1		101	G06F-015/00	CIP of application US 2001866101

Abstract (Basic): US 20040216030 A1
NOVELTY - The source extensible markup language (XML) schema and target XML schema are received and mapped to common ontology model. The transformation function for transforming data conforming to the source XML schema into data conforming to target XML schema, is derived by using the ontology model.
DETAILED DESCRIPTION - INDEPENDENT CLAIMS are also included for the following:
(1) transformation function deriving system;
(2) ontology model creating method;
(3) ontology model creating system;
(4) article of manufacture comprising computer readable medium storing **program** for transforming data from one schema to another; and
(5) article of manufacture comprising computer readable medium storing **program** for creating common ontology model.
USE - For deriving transformation function for transforming data related to physical thing such as car, boat, screw and transistor, and data related to relation between physical things such as personal computer (PC) and its flat panel screen, from one schema to another schema.
ADVANTAGE - The transformation function is derived easily.
DESCRIPTION OF DRAWING(S) - The figure shows a flowchart explaining transformation function deriving process.
pp; 101 DwgNo 1/26
Title Terms: TRANSFORM; FUNCTION; DERIVATIVE; METHOD; MAP; SOURCE; TARGET; COMMON; MODEL; TRANSFORM; DATA; CONFORM; SOURCE; DATA; CONFORM; TARGET
Derwent Class: T01
International Patent Class (Main): **G06F-015/00**
File Segment: EPI

17/5/8 (Item 5 from file: 350)
DIALOG(R)File 350:Derwent WPIX

015393649 **Image available**
WPI Acc No: 2003-455790/200343
XRPX Acc No: N03-362400
Operating system utility provision for developing application software, involves creating linkable objects from derived class and static function implementations, such that application program includes objects to create and use utility
Patent Assignee: COMPUTER ASSOC THINK INC (COMP-N)
Inventor: BIRZE B B
Number of Countries: 001 Number of Patents: 001
Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week	
US 6526457	B1	20030225	US 9629220	P	19961030	200343	B
			US 97957421	A	19971024		

Priority Applications (No Type Date): US 9629220 P 19961030; US 97957421 A 19971024
Patent Details:

Patent No	Kind	Lan	Pg	Main IPC	Filing Notes
US 6526457	B1		7	G06F-009/00	Provisional application US 9629220

Abstract (Basic): US 6526457 B1

NOVELTY - A derived class including base class, static and virtual **functions** to create **derived** class and static **function** implementations, is defined. The linkable objects are created from the implementations by compiling derived classes with flags and settings for specific operation system, such that application **program** includes file and objects to create and use operating system utility.

DETAILED DESCRIPTION - INDEPENDENT CLAIMS are also included for the following:

(1) application **program** creation method; and

(2) operating system accessing system.

USE - For providing operating system utility for use by application **program** , for developing application software for scheduling, input/output control, storage assignment, data management.

ADVANTAGE - Improves portability of application **programming** between computers having different operating systems, since the **program** in operating system is executed by linking the **program** with utility object library, thereby avoiding additional processes of compiling and linking to the system utilities. The application programmer easily invokes the utilities by learning only generic operation of service defined in base class without having knowledge of operating system.

DESCRIPTION OF DRAWING(S) - The figure illustrates relationship between utility object library and application **program** .

pp; 7 DwgNo 2/2

Title Terms: OPERATE; SYSTEM; UTILISE; PROVISION; DEVELOP; APPLY; SOFTWARE; LINK; OBJECT; DERIVATIVE; CLASS; STATIC; FUNCTION; APPLY; **PROGRAM** ; OBJECT; UTILISE
Derwent Class: T01
International Patent Class (Main): **G06F-009/00**
File Segment: EPI

17/5/10 (Item 7 from file: 350)
DIALOG(R)File 350:Derwent WPIX

014834137 **Image available**
WPI Acc No: 2002-654843/200270
XRPX Acc No: N02-517376
Computer program translator operation assuring method involves linking translated test procedures to portions of original program to produce test module

Patent Assignee: COMPAQ COMPUTER CORP (COPQ)
Inventor: ANDREWS K A; CUTLER D R; DEL VIGNA P; HERREN J L; MOLLOY M E
Number of Countries: 001 Number of Patents: 001
Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week	
US 6425118	B1	20020723	US 9752955	A	19970718	200270	B
			US 98118403	A	19980717		

Priority Applications (No Type Date): US 9752955 P 19970718; US 98118403 A 19980717
Patent Details:

Patent No	Kind	Lan	Pg	Main IPC	Filing Notes
US 6425118	B1		13	G06F-009/45	Provisional application US 9752955

Abstract (Basic): US 6425118 B1
NOVELTY - The test procedures that uses interface specification, obtained from **procedure** calls in original **program** , are **derived** . The translated test **procedures** are linked to portions of the original **program** to produce a test module which is executed to execute the procedures for determining whether the passed variable values are valid. An error condition is indicated if a passed value is invalid.
DETAILED DESCRIPTION - An INDEPENDENT CLAIM is included for computer **program** translator operation verifying system.
USE - For assuring the valid operations of translator e.g. Rosetta translator that translates computer **programs** from one language to another.
ADVANTAGE - Correct translation of text preprocessor mechanisms e.g. macros, conditionally compiled regions of **code** , and source file inclusion can be achieved. Automatically generates self-checking test for source-to-source translators to guarantee inter-language compatibility of interfaces.
DESCRIPTION OF DRAWING(S) - The figure shows an overview of automated interface test generator.
pp; 13 DwgNo 3/5
Title Terms: COMPUTER; **PROGRAM** ; TRANSLATION; OPERATE; ASSURE; METHOD; LINK; TRANSLATION; TEST; PROCEDURE; PORTION; ORIGINAL; **PROGRAM** ; PRODUCE ; TEST; MODULE
Derwent Class: T01
International Patent Class (Main): **G06F-009/45**
File Segment: EPI

17/5/11 (Item 8 from file: 350)
DIALOG(R)File 350:Derwent WPIX

014586525 **Image available**
WPI Acc No: 2002-407229/200244
XRPX Acc No: N02-319801
Programming **method for providing polymorphism in** programming **language, involves executing over-write** method **associated with** derived **class based on presence or absence of** method **over-write identifiers**
Patent Assignee: SUN MICROSYSTEMS INC (SUNM)
Inventor: JAHNKE J
Number of Countries: 026 Number of Patents: 002
Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week	
EP 1186996	A1	20020313	EP 2000119355	A	20000908	200244	B
US 20020152457	A1	20021017	US 2001269094	A	20010215	200270	
			US 2001949743	A	20010910		

Priority Applications (No Type Date): EP 2000119355 A 20000908
Patent Details:

Patent No	Kind	Lan	Pg	Main IPC	Filing Notes
EP 1186996	A1	E	32	G06F-009/44	

Designated States (Regional): AL AT BE CH CY DE DK ES FI FR GB GR IE IT LI LT LU LV MC MK NL PT RO SE SI
US 20020152457 A1 G06F-009/45 Provisional application US 2001269094

Abstract (Basic): EP 1186996 A1
NOVELTY - A method over-write identifier indicating over-written superior class **method** is associated with a **derived** class in which a **method** of superior class is being over-written. An over-write method associated with derived class is executed depending on pressure or absence of identifier in the derived class.
DETAILED DESCRIPTION - INDEPENDENT CLAIMS are also included for the following:
(a) Data processing device;
(b) Recorded medium storing polymorphism providing **program** ;
(c) Computer **program** for providing polymorphism
USE - For providing polymorphism in **programming** language.
ADVANTAGE - Provides polymorphism in any arbitrary **programming** language which by default does not provide polymorphism.
DESCRIPTION OF DRAWING(S) - The figure shows a flowchart explaining the **programming** method.
pp; 32 DwgNo 3/3
Title Terms: **PROGRAM** ; METHOD; POLYMORPH; **PROGRAM** ; LANGUAGE; EXECUTE; WRITING; METHOD; ASSOCIATE; DERIVATIVE; CLASS; BASED; PRESENCE; ABSENCE; METHOD; WRITING; IDENTIFY
Derwent Class: T01
International Patent Class (Main): **G06F-009/44** ; **G06F-009/45**
File Segment: EPI

17/5/13 (Item 10 from file: 350)
DIALOG(R)File 350:Derwent WPIX

014009391 **Image available**
WPI Acc No: 2001-493605/ **200154**
XRPX Acc No: N01-365492
Interpretation method of class dynamic link library used in application program , involves storing pointer of member function of derivation class which includes entity of virtual function , in a routine
Patent Assignee: NEC CORP (NIDE)
Number of Countries: 001 Number of Patents: 001
Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week	
JP 2001184215	A	20010706	JP 99368867	A	19991227	200154	B

Priority Applications (No Type Date): JP 99368867 A 19991227
Patent Details:

Patent No	Kind	Lan	Pg	Main IPC	Filing Notes
JP 2001184215	A		11	G06F-009/44	

Abstract (Basic): JP 2001184215 A
NOVELTY - The member function of class defined as a virtual function is called by executing the objective **program** . The pointer of the member function of the derivation class which includes the entity of the virtual function, is stored in routine (103).
USE - For interpretation of class dynamic link library.
ADVANTAGE - The entity of the virtual function defined in specified class of DLL, is callable and debug operation is simplified.
DESCRIPTION OF DRAWING(S) - The figure shows the block diagram showing the structure of the interpretor method of execution of the DLL class. (Drawing includes non-English language text).
Routine (103)
pp; 11 DwgNo 1/4
Title Terms: INTERPRETATION; METHOD; CLASS; DYNAMIC; LINK; LIBRARY; APPLY; **PROGRAM** ; STORAGE; POINT; MEMBER; FUNCTION; DERIVATIVE; CLASS; ENTITY;

VIRTUAL; FUNCTION; ROUTINE
Derwent Class: T01
International Patent Class (Main): **G06F-009/44**
File Segment: EPI

17/5/16 (Item 13 from file: 350)
DIALOG(R)File 350:Derwent WPIX

013450837 **Image available**
WPI Acc No: 2000-622780/ **200060**
XRPX Acc No: N00-461586
Software product failure detector acquires edition number of software products that correspond to failure based on derivation information, function list information and failure information
Patent Assignee: NEC CORP (NIDE)
Number of Countries: 001 Number of Patents: 001
Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week	
JP 2000250748	A	20000914	JP 9951980	A	19990226	200060	B

Priority Applications (No Type Date): JP 9951980 A 19990226
Patent Details:

Patent No	Kind	Lan	Pg	Main IPC	Filing Notes
JP 2000250748	A		6	G06F-009/06	

Abstract (Basic): JP 2000250748 A
NOVELTY - Information producing unit (M1) outputs derivation information of **program** . Function list information unit (M2) outputs function name and software product edition number. Failure information unit outputs information with **function** name during **program** failure. Based on **derivation** information, **function** list and failure information, edition number of software product corresponding to failure, is acquired.
USE - For detecting the version of software which failed when run by customer or version which needs improvement.
ADVANTAGE - Using the detector, the failed software version and the customer who detected the failure can be determined. Using failure information database, the error in software version can be detected.
DESCRIPTION OF DRAWING(S) - The figure shows the block diagram of software product.
Information producing unit (M1)
Function list information unit (M2)
pp; 6 DwgNo 1/5
Title Terms: SOFTWARE; PRODUCT; FAIL; DETECT; ACQUIRE; EDIT; NUMBER; SOFTWARE; PRODUCT; CORRESPOND; FAIL; BASED; DERIVATIVE; INFORMATION; FUNCTION; LIST; INFORMATION; FAIL; INFORMATION
Derwent Class: T01
International Patent Class (Main): **G06F-009/06**
International Patent Class (Additional): **G06F-011/34**
File Segment: EPI

17/5/17 (Item 14 from file: 350)
DIALOG(R)File 350:Derwent WPIX

013157126 **Image available**
WPI Acc No: 2000-328999/ **200028**
Related WPI Acc No: 2000-339151; 2000-339161; 2000-339162; 2001-490569; 2001-490570; 2002-654694; 2002-705322; 2003-584345; 2003-743191; 2004-345205
XRPX Acc No: N00-247680
Computer implemented system for performing data mining application,

includes analytic application programming **interface that generates set of scalable data mining functions**

Patent Assignee: NCR CORP (NATC)

Inventor: BRYE T M; HERMAN M H; MILLER T E; ROLLINS A L; TATE B D

Number of Countries: 089 Number of Patents: 004

Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week	
WO 200020997	A1	20000413	WO 99US22995	A	19991001	200028	B
AU 9964120	A	20000426	AU 9964120	A	19991001	200036	
EP 1208480	A1	20020529	EP 99951742	A	19991001	200243	
			WO 99US22995	A	19991001		
US 6438552	B1	20020820	US 98102831	P	19981002	200257	
			US 99411818	A	19991001		

Priority Applications (No Type Date): US 98102831 P 19981002; US 99411818 A 19991001

Patent Details:

Patent No	Kind	Lan	Pg	Main IPC	Filing Notes
WO 200020997	A1	E	31	G06F-017/30	

Designated States (National): AE AL AM AT AU AZ BA BB BG BR BY CA CH CN CR CU CZ DE DK DM EE ES FI GB GD GE GH GM HR HU ID IL IN IS JP KE KG KP KR KZ LC LK LR LS LT LU LV MD MG MK MN MW MX NO NZ PL PT RO RU SD SE SG SI SK SL TJ TM TR TT TZ UA UG US UZ VN YU ZA ZW

Designated States (Regional): AT BE CH CY DE DK EA ES FI FR GB GH GM GR IE IT KE LS LU MC MW NL OA PT SD SE SL SZ TZ UG ZW

AU 9964120	A			Based on patent WO 200020997
EP 1208480	A1	E	G06F-017/30	Based on patent WO 200020997

Designated States (Regional): DE FR GB

US 6438552	B1		G06F-017/30	Provisional application US 98102831

Abstract (Basic): WO 200020997 A1

NOVELTY - Relational database (116) is stored in data storage devices of computer. The analytic application **programming** interface (API) generates set of scalable data mining functions for performing data mining operations directly within the relational database management system (114) for accessing relational database stored in storage device.

DETAILED DESCRIPTION - The scalable data mining functions created by parametrizing and instantiating the analytic API process data collections stored in relational database and produce results that are stored in relational database. The scalable data mining functions are dynamically queries comprising combined phrases with substituting values based on parameters supplied to analytic API. The scalable data mining functions are selected from group of **functions** comprising data description **functions** , data **derivation functions** , data reduction **functions** , data reorganization **functions** , data sampling functions and data partitioning functions. INDEPENDENT CLAIMS are also included for the following:

(a) data mining application performing method;

(b) data mining application **programming program**

USE - For data mining assisting in relational database management system.

ADVANTAGE - Provides efficient usage of parallel processor computer system and provides foundation for data mining sets in relational database management system and hence allows data mining of large databases. Time saving can be gained by automating the desertion processes and insights from past experiences. Can be reused by saving and reapplying prior desertion technique.

DESCRIPTION OF DRAWING(S) - The figure shows the block diagram illustrating computer hardware environment.

Database management system (114)

Relational database (116)

pp; 31 DwgNo 1/5

Title Terms: COMPUTER; IMPLEMENT; SYSTEM; PERFORMANCE; DATA; MINE; APPLY; APPLY; PROGRAM ; INTERFACE; GENERATE; SET; DATA; MINE; FUNCTION

Derwent Class: T01

International Patent Class (Main): **G06F-017/30**
File Segment: EPI

17/5/18 (Item 15 from file: 350)
DIALOG(R)File 350:Derwent WPIX

010268678 **Image available**
WPI Acc No: 1995-169933/ **199522**
Related WPI Acc No: 1994-092703; 1998-311876; 1999-095189
XRPX Acc No: N95-133249

Generating object data structure layout for class in compiler for object-oriented programming **language - setting virtual function table pointer equal to address of pointer for class, else to address of virtual base table pointer, else to pointer of non-virtual base class, else to address of data member of class**

Patent Assignee: MICROSOFT CORP (MICR-N)
Inventor: JONES D T; O'RIORDAN M J; ZBIKOWSKI M J
Number of Countries: 001 Number of Patents: 001
Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week
US 5410705	A	19950425	US 91682537	A	19910409	199522 B
			US 93163846	A	19931207	

Priority Applications (No Type Date): US 91682537 A 19910409; US 93163846 A 19931207
Patent Details:

Patent No	Kind	Lan	Pg	Main IPC	Filing Notes
US 5410705	A		29	G06F-012/00	Div ex application US 91682537 Div ex patent US 5297284

Abstract (Basic): US 5410705 A

The method for a computer compiler for an object-oriented **programming** language for implementing virtual functions and virtual base classes uses the data structure layout of an object which includes a virtual function table pointer, a virtual base table pointer, occurrences of each non-virtual base class, the data members of the class, and occurrences of each virtual base class. If a class introduces a virtual function member and the class has a non-virtual base class with a virtual function table pointer, then the class shares the virtual function table pointer of the non-virtual base class that is first visited in a depth-first, left-to-right traversal of the inheritance tree.

Pref., each instance of a given class shares a set of virtual function tables and virtual base tables for that class. Adjusters are used when a **function** member in a **derived** class overrides a **function** member that is defined in more than one base class, and when a derived class has a base class that overrides a function member in a virtual base class of that class and the derived class itself does not override the function member.

ADVANTAGE - Reduces run-time storage requirements of each instance of class. Virtual function table pointers can be shared between derived and base classes. Reduces number of adjusters needed for virtual functions.

25/5/1 (Item 1 from file: 347)
DIALOG(R)File 347:JAPIO

07175686 **Image available**
ENCIPHERING SYSTEM AND DECIPHERING SYSTEM

PUB. NO.: 2002-044073 [JP 2002044073 A]
PUBLISHED: February 08, 2002 (20020208)
INVENTOR(s): TAKEDA KOJI
APPLICANT(s): CASIO COMPUT CO LTD
APPL. NO.: 2000-224534 [JP 2000224534]
FILED: July 25, 2000 (20000725)
INTL CLASS: H04L-009/26

ABSTRACT

PROBLEM TO BE SOLVED: To provide an enciphering/deciphering method, which improves reliability and can be applied to applications over a wide range.

SOLUTION: A device 10 generates a vector in the manner of chaos or successively while using a non-linear function for rotating and parallel advancing an n-dimensional (n 1) vector defined within the closed area of an n-dimensional space and enciphers a plane sentence M while using this vector. A cryptographic sentence C generated thereby is transmitted to a device 12. The device 12 generates the vector similarly to the device 10 and deciphers the cryptographic sentence C while using this vector. The devices 10 and 12 share a common key and the non-linear **function** is generated by a **parameter** corresponding to the **common** key. The non-linear **function** includes a rotary matrix for rotating the n-dimensional vector and this vector is successively generated so as not to be mutually matched.

25/5/2 (Item 2 from file: 347)
DIALOG(R)File 347:JAPIO

06317922 **Image available**
INFORMATION PROVIDING DEVICE AND METHOD

PUB. NO.: 11-259520 [JP 11259520 A]
PUBLISHED: September 24, 1999 (19990924)
INVENTOR(s): HIDAKA TETSUO
APPLICANT(s): NIPPON TELEGR & TELEPH CORP <NTT>
APPL. NO.: 10-063174 [JP 9863174]
FILED: March 13, 1998 (19980313)
INTL CLASS: G06F-017/30; G06F-015/00

ABSTRACT

PROBLEM TO BE SOLVED: To reduce the labor of a user to input a parameter at each time by preparing a function to identify and authenticate the user, a function to hold plural settings according to the user states and to automatically or manually select these settings, a function to change the parameter of each character, etc.

SOLUTION: A function is prepared to identify and authenticate a user together with a function to hold plural settings according to the user states which are set for each purpose using the retrieval result, each urgency and each communication environment and to automatically or manually select these settings, a function to change the parameter of each character, a **function** to **share** a variable **parameter** among characters,

and a **function** to extract the feature of each character from a usage history and to change these extracted features. In such a constitution of an information providing device, a parameter changing part 50 changes the parameter, a user recognizing part 60 recognizes the user, a character processing part 70 processes the character data and the parameter, and a user characteristic extracting part 100 extracts the user characteristic from the usage history.

25/5/4 (Item 4 from file: 347)
DIALOG(R)File 347:JAPIO

05678258 **Image available**
MULTIPROCESSOR COMPUTER SYSTEM PROVIDED WITH INTER-PROCESS MUTUAL EXCLUDING FUNCTION

PUB. NO.: 09-293058 [JP 9293058 A]
PUBLISHED: November 11, 1997 (19971111)
INVENTOR(s): YAMAUCHI MASAHIKO
YASHIRO HIROSHI
MURAYAMA HIDEKI
HORIKAWA KAZUO
HAYASHI TAKEHISA
YAMADA KIMITOSHI
APPLICANT(s): HITACHI LTD [000510] (A Japanese Company or Corporation), JP (Japan)
APPL. NO.: 08-106813 [JP 96106813]
FILED: April 26, 1996 (19960426)
INTL CLASS: [6] G06F-015/16; G06F-009/46
JAPIO CLASS: 45.4 (INFORMATION PROCESSING -- Computer Applications); 45.1 (INFORMATION PROCESSING -- Arithmetic Sequence Units)

ABSTRACT

PROBLEM TO BE SOLVED: To execute a succeeding processing without interrupting the execution of a process regardless of whether or not the instruction sequence of a danger area is executed by the process and to enhance use efficiency by providing a mutual excluding function.

SOLUTION: Locking is executed through the use of an inseparable memory reading means whereby memory reading and writing is executed by one instruction in order to exclusively operate data in a mutual exclusion data storing area (S804). Then, a counter variable is decreased by one in order to store the number of the processes by which mutual exclusion start procedure is called (S805). It is judged whether the value of the counter variable is negative or not (S806) and obtained locking is released. Then, the item of a process management table is retrieved through the use of a process identifier (S808) and the leading address of the area where the instruction string of danger area **procedure** which is designated as the **argument** of **mutual** exclusion start **procedure** is stored is set to signal procedure (S809).

25/5/5 (Item 5 from file: 347)
DIALOG(R)File 347:JAPIO

03108230 **Image available**
REALIZING SYSTEM FOR VARIABLE LENGTH DYNAMIC COMMON

PUB. NO.: 02-083730 [JP 2083730 A]
PUBLISHED: March 23, 1990 (19900323)
INVENTOR(s): MAEDA HIDEKO

YAMADA SHIGEMI
APPLICANT(s): HITACHI LTD [000510] (A Japanese Company or Corporation), JP (Japan)
APPL. NO.: 63-235088 [JP 88235088]
FILED: September 21, 1988 (19880921)
INTL CLASS: [5] G06F-009/06; G06F-009/45
JAPIO CLASS: 45.1 (INFORMATION PROCESSING -- Arithmetic Sequence Units)
JOURNAL: Section: P, Section No. 1063, Vol. 14, No. 283, Pg. 101, June 19, 1990 (19900619)

ABSTRACT

PURPOSE: To realize the more advantageous application and to improve the program performance in the title system by using a means which declares the common that defines a common name and the size of a memory area at compiling as a dynamic common and a means which decides the size of a dynamic common memory area.

CONSTITUTION: A main program PG 30 calls out the subprograms PG 30A and PG 30B and performs the due process. Prior to these calls a variable length dynamic common is declared with a common name and the securing **method** of a **common** area defined as the **parameters** , and a declaring **subroutine** DCOM is called out. A DCOM subroutine 311 secures all available memory areas and registers them into a dynamic common control table 34 as the dynamic areas of a common C1. Then the PG 30A is called and the dynamic common information 32 is taken out via an execution routine 33 for the start process. Thus an address constant which refers to the common C1 contained in a PG is resolved via a secured address. As a result, an area having the size secured in the routine 311 is available. So is with the PG 30B.

25/5/7 (Item 7 from file: 347)

02736032 **Image available**
CALLING CONTROL SYSTEM FOR SUBROUTINE

PUB. NO.: 01-033632 [JP 1033632 A]
PUBLISHED: February 03, 1989 (19890203)
INVENTOR(s): MITA HITOSHI
ITO YUTAKA
APPLICANT(s): PFU LTD [366680] (A Japanese Company or Corporation), JP (Japan)
APPL. NO.: 62-190996 [JP 87190996]
FILED: July 30, 1987 (19870730)
INTL CLASS: [4] G06F-009/40
JAPIO CLASS: 45.1 (INFORMATION PROCESSING -- Arithmetic Sequence Units)
JOURNAL: Section: P, Section No. 875, Vol. 13, No. 219, Pg. 50, May 23, 1989 (19890523)

ABSTRACT

PURPOSE: To simplify the processing operations carried out at the side of a **subroutine** by referring an **attribute** table via a calling control **common** program when the **subroutine** is called out and deciding based on said referring result whether a desired function is practicable or not.

CONSTITUTION: When a subroutine 2 or 3 is called out by a main program 1, a calling control common program 5 refers to a subroutine attribute table 4 to decide the relevant item based on the called subroutine, the specific function working presently and the present conditions. Then it is checked based on the obtained item whether the corresponding function executing subroutines 6-8 can be called out or not. Based on this check result, the corresponding one of those routines 6-8 is called out. Thus it is possible to simplify the process where it is checked whether a desired function

executing subroutine can be called out or not

25/5/8 (Item 1 from file: 350)
DIALOG(R)File 350:Derwent WPIX

014632042 **Image available**
WPI Acc No: 2002-452746/200248
XRPX Acc No: N02-356907
Multi-functional digital telephone - wherein the various functions share the same programming code and parameter
Patent Assignee: CHEN W (CHEN-I); JUNG C (JUNG-I)
Inventor: CHEN W; JUNG C
Number of Countries: 001 Number of Patents: 001
Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week	
TW 453099	A	20010901	TW 99123231	A	19991229	200248	B

Priority Applications (No Type Date): TW 99123231 A 19991229
Patent Details:

Patent No	Kind	Lan	Pg	Main IPC	Filing Notes
TW 453099	A			H04M-009/08	

Abstract (Basic): TW 453099 A
NOVELTY - A digital telephone is equipped with the following functions: message recording, on-line recording, voice dialing, tone adjustment, and voice pace adjustment. Moreover, each **function shares** its own programming code and **parameter** with other **functions**. For example, the parameter required for voice recognition can be obtained by transforming the parameter for voice compression. The functions of voice pace adjustment and tone adjustment share the same program. A method for processing digital voice signal is provided, which is applied in sampling and processing digitized voice data to enable the digital telephone equipped with the functions of tone adjustment and voice pace adjustment.
DwgNo 1/1
Title Terms: MULTI; FUNCTION; DIGITAL; TELEPHONE; VARIOUS; FUNCTION; SHARE; PROGRAM; CODE; PARAMETER
Derwent Class: W01; W04
International Patent Class (Main): H04M-009/08
International Patent Class (Additional): H04M-001/78; H04M-003/40
File Segment: EPI

25/5/10 (Item 3 from file: 350)
DIALOG(R)File 350:Derwent WPIX

014089770 **Image available**
WPI Acc No: 2001-573984/200165
XRPX Acc No: N01-427987
Data processor e.g. computer, generates information transmittance programs automatically for sharing parameter and functions between data analysis program and other programs
Patent Assignee: MATSUSHITA DENKI SANGYO KK (MATU)
Number of Countries: 001 Number of Patents: 001
Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week	
JP 2001142717	A	20010525	JP 99327408	A	19991117	200165	B

Priority Applications (No Type Date): JP 99327408 A 19991117
Patent Details:

Patent No	Kind	Lan	Pg	Main IPC	Filing Notes
JP 2001142717	A		15	G06F-009/45	

Abstract (Basic): JP 2001142717 A

NOVELTY - An analyzer analyzes the syntax of data analysis program input through an input unit (110). Based on the analysis result, the function of a data analysis program is determined. The parameter of this determined **function** is determined. The information transmittance program, for **sharing** the determined **parameter** and **function** between data analysis program and other program is generated automatically.

USE - E.g. computer for processing digital data, digital video signal, digital audio signal, input data of automatic controller, numeric data, etc.

ADVANTAGE - As the information transmittance program is generated automatically, the debug operation of information transmittance program is unnecessary. As the data analysis program is linked with other data analysis program, the functional extension of data processor is performed easily. Also parameters determined and displayed by the display device are highly reliable and a highly efficient data processor is obtained.

DESCRIPTION OF DRAWING(S) - The figure shows the block diagram of data process. (Drawing includes non-English language text).

Input unit (110)

pp; 15 DwgNo 1/6

Title Terms: DATA; PROCESSOR; COMPUTER; GENERATE; INFORMATION; TRANSMITTANCE; PROGRAM; AUTOMATIC; SHARE; PARAMETER; FUNCTION; DATA; ANALYSE; PROGRAM; PROGRAM

Derwent Class: T01

International Patent Class (Main): G06F-009/45

File Segment: EPI

?

26/5/1 (Item 1 from file: 347)
DIALOG(R)File 347:JAPIO

08093802 **Image available**
INHERITANCE PROCEDURE DIAGNOSIS SYSTEM AND **INHERITANCE PROCEDURE** DIAGNOSIS PROGRAM

PUB. NO.: 2004-206561 [JP 2004206561 A]
PUBLISHED: July 22, 2004 (20040722)
INVENTOR(s): HANDA MITSUGI
APPLICANT(s): SIGMA MANAGEMENT KK
APPL. NO.: 2002-376798 [JP 2002376798]
FILED: December 26, 2002 (20021226)
INTL CLASS: G06F-017/60

ABSTRACT

PROBLEM TO BE SOLVED: To construct a diagnosis system analyzing, displaying, and outputting an inheritance procedure and post mortem processing to be carried out by a successor when a householder dies according to occupation, a family composition, and an assets status of the householder.

SOLUTION: In a file device in this diagnosis system, an inheritance item file, which consists of **procedure** items of the **inheritance procedure** and its conditions, and an item contents file, which stores procedure contents such as a destination and a procedure limit of a procedure to be performed by the successor for each inheritance procedure item, are previously recorded. A user information input means receives input of user information, the user information is collated with information in the both files, and the inheritance procedure item to be carried out by the successor of a user and its contents are selected to be displayed and outputted as a certificate.

26/5/5 (Item 2 from file: 350)
DIALOG(R)File 350:Derwent WPIX

015000349 **Image available**
WPI Acc No: 2003-060864/200306
XRPX Acc No: N03-047042
Enabling access in object-oriented environment by instance of first class to selected protected resource of instance of second class by calling access-resource method inherited from third class
Patent Assignee: SUN MICROSYSTEMS INC (SUNM)
Inventor: KRIVORUCHKO M
Number of Countries: 027 Number of Patents: 002
Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week	
EP 1253514	A1	20021030	EP 2001303704	A	20010424	200306	B
US 20020199033	A1	20021226	US 2002131703	A	20020423	200308	

Priority Applications (No Type Date): EP 2001303704 A 20010424
Patent Details:

Patent No	Kind	Lan	Pg	Main IPC	Filing Notes
EP 1253514	A1	E	13	G06F-009/44	
	Designated States (Regional): AL AT BE CH CY DE DK ES FI FR GB GR IE IT LI LT LU LV MC MK NL PT RO SE SI TR				
US 20020199033	A1			G06F-009/44	

Abstract (Basic): EP 1253514 A1

NOVELTY - A second class is defined as a subclass of a third class such that the second class inherits from the third class and implements the protected virtual method for accessing the selected protected resource by overloading that method. An instance of the first class calling the access-resource **method inherited** from the third class may access-resource **method** using the pointer to the instance of the second class and the overloaded method to access the resource.

DETAILED DESCRIPTION - INDEPENDENT CLAIMS are included for:
(a) a computer program product
(b) a computer system

USE - In an object-oriented environment, the resources provided by instances of classes can be defined as being public (in which case other instances of other classes can have access to them), or can be defined as protected (in which case other instances of other classes cannot normally access them).

ADVANTAGE - Enables one class to access a protected resource of the other class by arranging that both classes inherit from the third class, which forms a handshake (or service handle) class.

DESCRIPTION OF DRAWING(S) - The drawing is a schematic block diagram of software components of an embodiment of the invention.

pp; 13 DwgNo 3/4

Title Terms: ENABLE; ACCESS; OBJECT; ORIENT; ENVIRONMENT; INSTANCE; FIRST; CLASS; SELECT; PROTECT; RESOURCE; INSTANCE; SECOND; CLASS; CALL; ACCESS; RESOURCE; METHOD; INHERITED; THIRD; CLASS
Derwent Class: T01
International Patent Class (Main): G06F-009/44
File Segment: EPI

26/5/8 (Item 5 from file: 350)
DIALOG(R)File 350:Derwent WPIX

013483198 **Image available**
WPI Acc No: 2000-655141/200063
Related WPI Acc No: 1998-322225
XRPX Acc No: N00-485558

Object management method in object oriented programming environment, involves performing alteration to content of object, at run time only after predetermined identification

Patent Assignee: INT BUSINESS MACHINES CORP (IBMC)
Inventor: BRESLAU F C; GREENSTEIN P G; RODELL J T
Number of Countries: 001 Number of Patents: 001
Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week
US 6101501	A	20000808	US 95579543	A	19951227	200063 B
			US 9853542	A	19980401	

Priority Applications (No Type Date): US 95579543 A 19951227; US 9853542 A 19980401
Patent Details:

Patent No	Kind	Lan	Pg	Main IPC	Filing Notes
US 6101501	A		12	G06F-017/30	Cont of application US 95579543 Cont of patent US 5758349

Abstract (Basic): US 6101501 A

NOVELTY - The method involves identifying alteration to content of an object, which includes information regarding execution method of certain function and a datum. Then, further alteration to the content of the object, is performed at run time .

DETAILED DESCRIPTION - An INDEPENDENT CLAIM is also included for object management system.

USE - For modifying an object through inheritance and disinheritance of methods and data at run-time, in object-oriented programming environment.

ADVANTAGE - Single object can have greater versatility, since it adapts to its changing run-time environment by **inheriting** and disinheriting **methods** and data. **Inheriting methods** and data into the object when needed and disinheriting them when no longer required, minimizes the size of object. Enables effective utilization of system resources by selective inheritance of data and methods into object.

DESCRIPTION OF DRAWING(S) - The figure shows the flow diagram of process run time inheritance.

```
File   8:Ei Compendex(R) 1970-2005/Sep W3
         (c) 2005 Elsevier Eng.  Info. Inc.
File  35:Dissertation Abs Online 1861-2005/Aug
         (c) 2005 ProQuest Info&Learning
File  65:Inside Conferences 1993-2005/Sep W4
         (c) 2005 BLDSC all rts. reserv.
File   2:INSPEC 1969-2005/Sep W3
         (c) 2005 Institution of Electrical Engineers
File  94:JICST-EPlus 1985-2005/Jul W5
         (c)2005 Japan Science and Tech Corp(JST)
File   6:NTIS 1964-2005/Sep W3
         (c) 2005 NTIS, Intl Cpyrght All Rights Res
File 144:Pascal 1973-2005/Sep W3
         (c) 2005 INIST/CNRS
File 434:SciSearch(R) Cited Ref Sci 1974-1989/Dec
         (c) 1998 Inst for Sci Info
File  34:SciSearch(R) Cited Ref Sci 1990-2005/Sep W3
         (c) 2005 Inst for Sci Info
File  99:Wilson Appl. Sci & Tech Abs 1983-2005/Jul
         (c) 2005 The HW Wilson Co.
File 266:FEDRIP 2005/Jun
         Comp & dist by NTIS, Intl Copyright All Rights Res
File  95:TEME-Technology & Management 1989-2005/Aug W3
         (c) 2005 FIZ TECHNIK

Set     Items   Description
S1       3854   FUNCTION? ?(7X)(DERIV??? OR DERIVATION OR INHERIT?)(7X)FUN-
            CTION? ?
S2        288   PROCEDURE? ?(7X)(DERIV??? OR DERIVATION OR INHERIT?)(7X)PR-
            OCEDURE? ?
S3       4038   METHOD? ?(7X)(DERIV??? OR DERIVATION OR INHERIT?)(7X)METHO-
            D? ?
S4      36526   (SHAR??? OR COMMON OR MUTUAL??)(5N)(ATTRIBUTE? ? OR ARGUME-
            NT? ? OR PROPERTY OR PROPERTIES OR PARAMETER? ?)
S5         51   FUNCTION? ?(7X)S4(7X)FUNCTION? ?
S6          9   PROCEDURE? ?(7X)S4(7X)PROCEDURE? ?
S7         88   METHOD? ?(7X)S4(7X)METHOD? ?
S8        880   (S1:S3 OR S5:S7) AND (PROGRAM? ? OR PROGRAMMING OR CODE? ?
            OR CODING OR INSTRUCTIONS)
S9         87   S8 AND INHERIT?
S10        63   RD (unique items)
S11        57   S10 NOT PY=2002:2005
S12        23   S8 AND S4
S13        15   RD (unique items)
S14        13   S13 NOT S10
S15     36338   SUBROUTINE? ? OR SUBPROGRAM? ? OR SUB()(ROUTINE? ? OR PROG-
            RAM? ?)
S16         0   S15(7X)(DERIV??? OR DERIVATION OR INHERIT?)(7X)S15
```

11/5/3 (Item 3 from file: 8)
DIALOG(R)File 8:Ei Compendex(R)

05083524 E.I. No: EIP98084312499
Title: Object-oriented development based on polymorphism patterns and optimization to reduce executable code size
Author: Naya, Hidemitsu; Narisawa, Fumio; Yokoyama, Takanori; Ohkawa, Keiichiro; Amano, Matsuo
Corporate Source: Hitachi Ltd, Ibaraki-ken, Jpn
Conference Title: Proceedings of the 1997 Conference on Technology of Object-Oriented Languages and Systems
Conference Location: Melbourne, Aust Conference Date: 19971124-19971128
Sponsor: Interactive Software Eng., Inc
E.I. Conference No.: 48734
Source: TOOLS 25 Proceedings of the Conference on Technology of Object-Oriented Languages and Systems, TOOLS 1997. IEEE Comp Soc, Los Alamitos, CA, USA. p 68-78
Publication Year: 1997
CODEN: 002837
Language: English
Document Type: CA; (Conference Article) Treatment: T; (Theoretical)
Journal Announcement: 9809W5
Abstract: This paper describes an object-oriented development method and an optimization method for embedded control systems. In embedded control systems development, specifications are changed frequently and there is strong constraint of memory. We present an object-oriented analysis and design method based on polymorphism patterns. Polymorphism patterns are standard of a method interfaces which are shared by several objects. With this method, a system is constructed with objects which have polymorphism patterns. This system ensures reusability because it easy to replace objects where the specification of the system is changed. Object-oriented technology has several **functions** , such as instantiation, **inheritance** and polymorphism, where **functions** are implemented with both method-tables and **inheritance** hierarchy tables. But these mechanisms are needless, in the automotive engine control application which execute fixed control flow. Our optimization method eliminates these mechanisms and reduces executable **code** size. We have applied above techniques to the development of automotive engine control applications. (Author abstract) 8 Refs.
Descriptors: *Object oriented **programming** ; Computer architecture; Constraint theory; Data storage equipment; Computer hardware description languages; Interfaces (computer); Optimization; **Codes** (symbols); Computer systems **programming**
Identifiers: Embedded control systems; Polymorphism patterns; Executable **code** sizes
Classification Codes:
723.1.1 (Computer Programming Languages)
723.1 (Computer Programming); 721.1 (Computer Theory, Includes Formal Logic, Automata Theory, Switching Theory, Programming Theory); 722.1 (Data Storage, Equipment & Techniques)
723 (Computer Software); 722 (Computer Hardware); 721 (Computer Circuits & Logic Elements)
72 (COMPUTERS & DATA PROCESSING)

11/5/4 (Item 4 from file: 8)
DIALOG(R)File 8:Ei Compendex(R)

04882053 E.I. No: EIP97123945477
Title: Parallel multifrontal algorithm and its implementation
Author: Geng, P.; Oden, J.T.; van de Geijn, R.A.
Corporate Source: Univ of Texas at Austin, Austin, TX, USA
Conference Title: Proceedings of the 1997 Symposium on Advances in Computational Mechanics. Part 1 (of 3)

Conference Location: Austin, TX, USA Conference Date: 19970112-19970115
E.I. Conference No.: 47344
Source: Computer Methods in Applied Mechanics and Engineering v 149 n 1-4 Oct 1997. p 289-301
Publication Year: 1997
CODEN: CMMECC ISSN: 0045-7825
Language: English
Document Type: JA; (Journal Article) Treatment: T; (Theoretical)
Journal Announcement: 9801W4
Abstract: In this paper, we describe a multifrontal method for solving sparse systems of linear equations arising in finite element and finite difference methods. The method proposed in this study is a combination of the nested dissection ordering and the frontal method. It can significantly reduce the storage and computational time required by the conventional direct methods and is also a natural parallel algorithm. In addition, the **method inherits** major advantages of the frontal **method** , which include a simple interface with finite element **codes** and an effective data structure so that the entire computation is performed element by element on a series of small linear systems with dense stiffness matrices. The numerical implementation targets both distributed-memory machines as well as conventional sequential machines. Its performance is tested through a series of examples. (Author abstract) 11 Refs.
Descriptors: *Parallel algorithms; Finite element method; Finite difference method; Computational complexity; Data structures; Stiffness matrix; Distributed computer systems; Sequential machines
Identifiers: Sparse linear equation systems
Classification Codes:
921.6 (Numerical Methods); 721.1 (Computer Theory, Includes Formal Logic, Automata Theory, Switching Theory, Programming Theory); 723.2 (Data Processing); 921.1 (Algebra); 722.4 (Digital Computers & Systems)
921 (Applied Mathematics); 721 (Computer Circuits & Logic Elements); 723 (Computer Software); 722 (Computer Hardware)
92 (ENGINEERING MATHEMATICS); 72 (COMPUTERS & DATA PROCESSING)

11/5/5 (Item 5 from file: 8)
DIALOG(R)File 8:Ei Compendex(R)

04623914 E.I. No: EIP97023518478
Title: Automatic inheritance **hierarchy restructuring and method refactoring**
Author: Moore, Ivan
Corporate Source: Univ of Manchester, Manchester, UK
Conference Title: Proceedings of the 1996 Conference on Object-Oriented Programming Systems, Languages & Applications
Conference Location: San Jose, CA, USA Conference Date: 19961006-19961010
Sponsor: ACM SIGPLAN
E.I. Conference No.: 45972
Source: SIGPLAN Notices (ACM Special Interest Group on Programming Languages) v 31 n 10 Oct 1996. ACM, New York, NY, USA. p 235-250
Publication Year: 1996
CODEN: SINODQ ISSN: 0362-1340
Language: English
Document Type: CA; (Conference Article) Treatment: G; (General Review); T; (Theoretical)
Journal Announcement: 9704W1
Abstract: Most object-oriented **programs** have imperfectly designed **inheritance** hierarchies and imperfectly factored methods, and these imperfections tend to increase with maintenance. Hence, even object-oriented **programs** are more expensive to maintain, harder to understand and larger than necessary. Automatic restructuring of **inheritance** hierarchies and refactoring of **methods** can improve the design of **inheritance** hierarchies, and the factoring of **methods** . This

results in **programs** being smaller, having better **code** re-use and being more consistent. This paper describes Guru, a prototype tool for automatic **inheritance** hierarchy restructuring and method refactoring of Self **programs** . Results from realistic applications of the tool are presented. (Author abstract) 21 Refs.

Descriptors: *Object oriented **programming** ; Computer aided software engineering; Software prototyping; Computer software

Identifiers: Automatic **inheritance** hierarchy restructuring; **Inheritance** hierarchies

Classification Codes:

723.1 (Computer Programming); 723.5 (Computer Applications)

723 (Computer Software)

72 (COMPUTERS & DATA PROCESSING)

11/5/6 (Item 6 from file: 8)

DIALOG(R)File 8:Ei Compendex(R)

04560890 E.I. No: EIP96113416155

Title: Fuzzy classes in object-oriented logic programming

Author: Baldwin, J.F.; Martin, T.P.

Corporate Source: Univ Walk, Bristol, UK

Conference Title: Proceedings of the 1996 5th IEEE International Conference on Fuzzy Systems. Part 2 (of 3)

Conference Location: New Orleans, LA, USA Conference Date: 19960908-19960911

Sponsor: IEEE

E.I. Conference No.: 45586

Source: IEEE International Conference on Fuzzy Systems v 2 1996. IEEE, Piscataway, NJ, USA,96CB35998. p 1358-1364

Publication Year: 1996

CODEN: PIFSFZ

Language: English

Document Type: CA; (Conference Article) Treatment: A; (Applications); T ; (Theoretical)

Journal Announcement: 9701W2

Abstract: Object-oriented **programming** has been widely adopted as a powerful **programming** paradigm, enabling software engineers to design systems using structures which map naturally onto the problem domain. Using ideas from logic **programming** , a class definition can be treated as a logic theory, making it amenable to formal reasoning. Each class corresponds to a set of objects in the problem domain. We extend this idea, so that an object may have a degree of membership in more than one class, i.e. each class represents a fuzzy set of objects in the problem domain. Uncertainty is also allowed in data values and computational **methods** associated with objects. The problem of multiple **inheritance** is addressed by including **methods** for resolving conflict in class definitions. A system is being implemented in Fril, to allow development of fuzzy object oriented knowledge-based applications. This system is known as Fril plus plus . The ideas are also applicable to other logic **programming** systems, assuming a method can be programmed at the object or meta-level to deal with uncertainty in terms, relations, and inference. (Author abstract) 15 Refs.

Descriptors: *Logic **programming** ; Object oriented **programming** ; Formal logic; Fuzzy sets; Membership functions; Calculations; Knowledge based systems

Identifiers: Fuzzy classes

Classification Codes:

723.4.1 (Expert Systems)

723.1 (Computer Programming); 723.4 (Artificial Intelligence)

723 (Computer Software); 921 (Applied Mathematics)

72 (COMPUTERS & DATA PROCESSING); 92 (ENGINEERING MATHEMATICS)

11/5/7 (Item 7 from file: 8)
DIALOG(R)File 8:Ei Compendex(R)

04493659 E.I. No: EIP96093327568
Title: Evaluation of dynamic object inheritance **in distributed environments**
Author: Kavi, Krishna; Wyatt, Barbara; Shirazi, Behrooz
Corporate Source: Univ of Texas at Arlington, Arlington, TX, USA
Source: Microcomputer Applications v 15 n 1 1996. p 26-37
Publication Year: 1996
CODEN: MIAPEZ ISSN: 0820-0750
Language: English
Document Type: JA; (Journal Article) Treatment: G; (General Review); T; (Theoretical)
Journal Announcement: 9611W1
Abstract: In this paper, we compare several approaches to the implementation of dynamic **inheritance** in distributed processing environments. We study approaches based on the use of remote procedure calls, futures, and asynchronous procedure calls. We also investigate dynamic linking of **inherited methods** , use of a `working-set' of **inherited methods** , and process migration to implement **inheritance** in distributed processing. The performances of these alternative approaches depend on the cost of communication, average **method** size, frequency of **inherited method** calls, and network traffic. All of these approaches incur run-time overhead. We feel that dynamic **inheritance** affords more flexibility and capability (than static **inheritance**), and justifies any overhead due to run-time support. (Author abstract) 2 Refs.
Descriptors: *Object oriented **programming** ; Distributed computer systems ; Parallel processing systems; Data communication systems; Telecommunication traffic
Identifiers: Remote procedure calls; Asynchronous procedure calls; Working sets; Dynamic **inheritance** ; Network traffic; Run time support
Classification Codes:
723.1 (Computer Programming); 722.4 (Digital Computers & Systems)
723 (Computer Software); 722 (Computer Hardware); 716 (Radar, Radio & TV Electronic Equipment)
72 (COMPUTERS & DATA PROCESSING); 71 (ELECTRONICS & COMMUNICATIONS)

11/5/8 (Item 8 from file: 8)
DIALOG(R)File 8:Ei Compendex(R)

04267698 E.I. No: EIP95102895271
Title: I plus : a multiparadigm language for object-oriented declarative** programming
Author: Ng, K.W.; Luk, C.K.
Corporate Source: Chinese Univ of Hong Kong, Shatin, Hong Kong
Source: Computer Languages v 21 n 2 Jul 1995. p 81-100
Publication Year: 1995
CODEN: COLADA ISSN: 0096-0551
Language: English
Document Type: JA; (Journal Article) Treatment: G; (General Review)
Journal Announcement: 9512W2
Abstract: This paper presents a multiparadigm language I** plus which is an integration of the three major **programming** paradigms: object-oriented, logic and functional. I** plus has an object-oriented framework in which the notions of classes, objects, **methods** , **inheritance** and message passing are supported. **Methods** may be specified as clauses or functions, thus the two declarative paradigms are incorporated at the method level of the object-oriented paradigm. In addition, two levels of parallelism may be exploited in I** plus **programming** . Therefore I** plus is a multiparadigm language for object-oriented declarative **programming** as well as parallel **programming** . (Author abstract) 41 Refs.

Descriptors: *Object oriented **programming** ; PROLOG (**programming** language); Computer aided software engineering; Logic **programming** ; Parallel processing systems; Computer aided design; Computer software; Real time systems; Distributed computer systems
Identifiers: Multiparadigm language; Functional paradigm; Declarative **programming**
Classification Codes:
723.1.1 (Computer Programming Languages)
723.1 (Computer Programming); 723.5 (Computer Applications); 722.4 (Digital Computers & Systems)
723 (Computer Software); 722 (Computer Hardware)
72 (COMPUTERS & DATA PROCESSING)

11/5/10 (Item 10 from file: 8)
DIALOG(R)File 8:Ei Compendex(R)

03064609 E.I. Monthly No: EIM9105-020929
Title: Method Schemas.
Author: Abiteboul, Serge; Kanellakis, Paris C.; Waller, Emmanuel
Corporate Source: I.N.R.I.A., Fr
Conference Title: Proceedings of the 9th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems
Conference Location: Nashville, TN, USA Conference Date: 19900402
Sponsor: SIGACT; SIGMOD; SIGART
E.I. Conference No.: 14153
Source: Proceedings of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems n 2-4. Publ by ACM, New York, NY, USA. p 16-27
Publication Year: 1990
CODEN: PSDSEQ
Language: English
Document Type: PA; (Conference Paper) Treatment: T; (Theoretical); X; (Experimental)
Journal Announcement: 9105
Abstract: The concept of method schemas is proposed as a simple model for object-oriented **programming** with features such as classes with **methods** and **inheritance** , **method** name overloading, and late binding. An important issue is to check whether a given method schema can possibly lead to inconsistencies in some interpretations. The consistency problem for method schemas is studied. The problem is shown to be undecidable in general. Decidability is obtained for monadic and/or recursion-free method schemas. The effect of covariance is considered. The issues of incremental consistency checking and of a sound algorithm for the general case are briefly discussed. (Author abstract) 20 Refs.
Descriptors: *COMPUTER **PROGRAMMING** --*Object Oriented **Programming** ; DATABASE SYSTEMS--Research; COMPUTER **PROGRAMMING** --Object Oriented **Programming**
Identifiers: METHOD SCHEMAS
Classification Codes:
723 (Computer Software)
72 (COMPUTERS & DATA PROCESSING)

11/5/18 (Item 5 from file: 2)
DIALOG(R)File 2:INSPEC

07315188 INSPEC Abstract Number: C1999-09-6110J-029
Title: A method of class decomposition based on inheritance
Author(s): Gu Zhi-Min; Kang Ji-Chang; Zheng Shou-Qi
Author Affiliation: Dept. of Comput. & Eng., Northwestern Polytech. Univ., Xian, China
Journal: Chinese Journal of Computers vol.21, no.11 p.1033-6

Publisher: Science Press,
Publication Date: Nov. 1998 Country of Publication: China
CODEN: JIXUDT ISSN: 0254-4164
SICI: 0254-4164(199811)21:11L.1033:MCDB;1-H
Material Identity Number: B714-1999-007
Language: Chinese Document Type: Journal Paper (JP)
Treatment: Practical (P)
Abstract: If the delegation mechanism is used in the distributed computing of objects, the communication overhead for **inheritance** is very high. In this paper, a **method** of class decomposition based on **inheritance** is given. When the **method** is used in OOP, the **inherited** relations of classes can be kept. The given method is evaluated on the platform of the ParCLOS system and shows high performance distributed computing. (4 Refs)
Subfile: C
Descriptors: distributed object management; **inheritance** ; object-oriented **programming** ; software performance evaluation
Identifiers: class decomposition; **inheritance** ; delegation mechanism; distributed object computing; communication overhead; object oriented **programming** ; ParCLOS system; performance
Class Codes: C6110J (Object-oriented programming); C6150N (Distributed systems software); C6120 (File organisation)

11/5/19 (Item 6 from file: 2)
DIALOG(R)File 2:INSPEC

06899652 INSPEC Abstract Number: C9806-6110J-002
Title: Fuzzy objects and multiple inheritance **in Fril++**
Author(s): Baldwin, J.F.; Martin, T.P.
Author Affiliation: Dept. of Eng. Math., Bristol Univ., UK
Conference Title: Fourth European Congress on Intelligent Techniques and Soft Computing Proceedings, EUFIT '96 Part vol.1 p.680-4 vol.1
Publisher: Verlag Mainz, Aachen, Germany
Publication Date: 1996 Country of Publication: Germany 3 vol. xxx+2298 pp.
ISBN: 3 89653 187 5 Material Identity Number: XX96-02141
Conference Title: Proceedings of Fourth European Congress on Intelligent Techniques and Soft Computing (EUFIT'96)
Conference Date: 2-5 Sept. 1996 Conference Location: Aachen, Germany
Availability: Verlag Mainz, Susterfeldstrasse 83, D-52072 Aachen, Germany
Language: English Document Type: Conference Paper (PA)
Treatment: Practical (P)
Abstract: The combination of logic **programming** , uncertainty, and object oriented methods in the Fril++ system promises much for the knowledge engineer dealing with real world applications. Object oriented **programming** and design methods currently occupy a central role in software engineering, partly because of software reuse and modularity, but also because classes correspond to identifiable categories in the problem domain. Using ideas from logic **programming** , a class definition can be treated as a logic theory. Each class corresponds to a set of objects in the problem domain. We extend this idea, so that an object may have a degree of membership in more than one class, i.e. each class represents a fuzzy set of objects in the problem domain. Uncertainty is also allowed in data values and computational **methods** associated with objects. The problem of multiple **inheritance** is addressed by including **methods** for resolving conflict in class definitions. (16 Refs)
Subfile: C
Descriptors: fuzzy set theory; inference mechanisms; **inheritance** ; knowledge based systems; logic **programming** ; object-oriented languages; object-oriented **programming** ; uncertainty handling
Identifiers: fuzzy objects; multiple **inheritance** ; Fril++; logic **programming** ; uncertainty; object oriented methods; real world applications

; object oriented **programming** ; software reuse; identifiable categories; class definition; logic theory; fuzzy set; problem domain; data values; computational methods; conflict resolution; class definitions; knowledge engineering
Class Codes: C6110J (Object-oriented programming); C6110L (Logic programming); C6170K (Knowledge engineering techniques); C6120 (File organisation); C1160 (Combinatorial mathematics); C4210 (Formal logic)

11/5/37 (Item 24 from file: 2)
DIALOG(R)File 2:INSPEC

03604073 INSPEC Abstract Number: C86014048
Title: Generalized inheritance
Author(s): Marcus, R.
Author Affiliation: Knowledge Syst. Lab., Hewlett-Packard Co., Cupertino, CA, USA
Journal: SIGPLAN Notices vol.20, no.11 p.47-8
Publication Date: Nov. 1985 Country of Publication: USA
CODEN: SINODQ ISSN: 0362-1340
Language: English Document Type: Journal Paper (JP)
Treatment: Practical (P)
Abstract: This article introduces a new paradigm for structuring **programs** called generalized **inheritance** . It is very elegant and easy to understand while possessing great potential power. Generalized **inheritance** is based on the concepts of object-oriented **programming** . Objects, sometimes called flavors or classes, are essentially data structures with attached functions and procedures collectively called methods. **Programming** is done by sending messages to instances of objects requesting the performance of some method. It is possible to define sub-objects of objects and attach new **methods** to the sub-objects. **Inheritance** is a mechanism by which all the **methods** of an object are automatically available to their sub-objects. The combination of objects and **inheritance** permits modularity while avoiding the redundant **coding** of methods. (0 Refs)
Subfile: C
Descriptors: structured **programming**
Identifiers: structuring **programs** ; generalized **inheritance** ; object-oriented **programming** ; data structures; redundant **coding**
Class Codes: C6110 (Systems analysis and programming)

11/5/52 (Item 2 from file: 99)
DIALOG(R)File 99:Wilson Appl. Sci & Tech Abs

1540689 H.W. WILSON RECORD NUMBER: BAST97055265
The sensible use of method overriding
Papurt, David M; LeJacq, Jean Pierre
Journal of Object-Oriented Programming v. 10 (Mar./Apr. '97) p. 62-5+
DOCUMENT TYPE: Feature Article ISSN: 0896-8438 LANGUAGE: English
RECORD STATUS: New record

ABSTRACT: Concrete guidelines for the sensible application of **inheritance** and method overriding are presented. Polymorphism provides an understandable and flexible method body selection framework, thereby simplifying reasoning about **program** behavior. All descendants **inherit** the interface for final, nonabstract, and abstract **methods** , but **inheritance** of implementation changes with the different **methods** . The substitution principle **programming** convention clarifies reasoning about **inherited** functionality and overall **program** behavior, but to obtain the full benefits of this convention, the overriding method should be in accordance with fairly constraining paradigmatic forms.

DESCRIPTORS: Polymorphic **programming** languages; **Inheritance** rule (Computers); Abstraction principle (Computers;

11/5/53 (Item 1 from file: 95)

01117853 I97072459279
Enhanced reusability via polymorphic additive virtual methods in C++
(Fortgeschrittene Wiederverwendbarkeit mit polymorphen additiven virtuellen Methoden in C++)
Chattamvelli, R; Saiedian, H
Dept. of Comput. Sci., Nebraska Univ., Omaha, NE, USA
Information and Software Technology, v39, n6, pp403-415, 1997
Document type: journal article Language: English
Record type: Abstract
ISSN: 0950-5849

ABSTRACT:
The virtual method is a useful concept in polymorphic behavior of object-oriented **programs** . By making a method virtual in a class, all classes derived from that class are allowed to modify or enhance the definition of the method (while retaining its original signature) providing one kind of polymorphism. We explore the virtues of virtual methods and introduce different ways of implementing additive virtual methods in C++. The concepts presented can find applications in shared software libraries, integrating software applications, and distributed computing.
DESCRIPTORS: C-- **PROGRAMMING** LANGUAGE; OBJECT ORIENTED **PROGRAMMING** ; ALLOTROPY; DISTRIBUTED COMPUTING; **PROGRAMMING** LANGUAGES; REUSABILITY; **PROGRAM** LIBRARY; APPLICATION SOFTWARE; OBJECT ORIENTED LANGUAGES; **PROGRAM** REUSABILITY
IDENTIFIERS: **INHERITANCE** ; ENHANCED REUSABILITY; POLYMORPHIC ADDITIVE VIRTUAL METHODS; OBJECT ORIENTED **PROGRAMS** ; SHARED SOFTWARE LIBRARIES; INTEGRATING SOFTWARE APPLICATIONS; ASSISTANT **FUNCTIONS** ; LATE BINDING; MULTIPLE **INHERITANCE** ; VIRTUAL TABLES; VIRTUAL MEMBER **FUNCTIONS** ; VERERBUNG; objektorientierte Programmierung; Vererbung; Polymorphie

11/5/54 (Item 2 from file: 95)

00978524 I96034596259
Typechecking and modules for multimethods
(Typpruefung und Module fuer Multimethoden)
Chambers, C; Leavens, GT
Dept. of Comput. Sci. & Eng., Washington Univ., Seattle, WA, USA
ACM Transactions on Programming Languages and Systems, v17, n6, pp805-843, 1995
Document type: journal article Language: English
Record type: Abstract
ISSN: 0164-0925

ABSTRACT:
Two major obstacles that hinder the wider acceptance of multimethods are: (1) concerns over the lack of encapsulation and modularity, and (2) the absence of static type-checking in existing multimethod-based languages. This article addresses both of these problems. We present a polynomial-time, static type-checking algorithm that checks the conformance, completeness and consistency of a group of method implementations with respect to declared message signatures. This algorithm improves on previous algorithms by handling separate type and **inheritance** hierarchies, abstract classes and graph-based method lookup semantics. We also present a module system that enables independently developed **code** to

be fully encapsulated and statically type-checked on a per-module basis. To guarantee that potential conflicts between independently developed modules have been resolved, a simple well-formedness condition on the modules comprising a **program** is checked at link-time. The type-checking algorithm and module system are applicable to a range of multimethod-based languages, but the article uses the Cecil language as a concrete example of how they can be applied.